PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Reconfiguração Dinâmica de Projetos de Software: Um modelo para integrar a gerência de projetos de software com os fluxos organizacionais

Maurício Covolan Rosito

Tese de Doutorado apresentada como requisito parcial para obtenção do grau de Doutor em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Ricardo Melo Bastos

PORTO ALEGRE
2014

Dados Internacionais de Catalogação na Publicação (CIP)

R821r Rosito, Maurício Covolan
Reconfiguração dinâmica de projetos de software : um modelo para integrar a gerência de projetos de software com os fluxos organizacionais / Maurício Covolan Rosito. – Porto Alegre, 2014.
247 p.

Tese (Doutorado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Ricardo Melo Bastos.

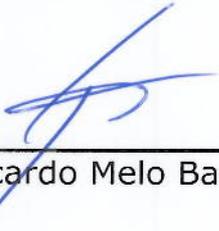
1. Informática. 2. Engenharia de Software. 3. Administração de Projetos. I. Bastos, Ricardo Melo. II. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE TESE DE DOUTORADO

Tese intitulada "Reconfiguração Dinâmica de Projetos de Software: Um Modelo para Integrar a Gerência de Projetos de Software com os Fluxos Organizacionais", apresentada por Maurício Covolan Rosito, como parte dos requisitos para obtenção do grau de Doutor em Ciência da Computação, aprovada em 23/01/2014 pela Comissão Examinadora:

Prof. Dr. Ricardo Melo Bastos – Orientador — PPGCC/PUCRS

Prof. Dr. Rafael Prikladnicki - — PPGCC/PUCRS

Profa. Dra. Edimara Mezzomo Luciano - — PPGAD/PUCRS

Prof. Dr. José Palazzo Moreira de Oliveira – — UFRGS

Homologada em...../....../......., conforme Ata No. .... pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

Campus Central
Av. Ipiranga, 6681 - P. 32 - sala 507 - CEP: 90619-900
Fone: (51) 3320-3611 - Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

AGRADECIMENTOS

A construção de uma tese de doutorado certamente é uma das tarefas mais desafiadoras einteressantes da vida acadêmica. Devido ao alto grau de dedicação e envolvimento exigidos, somente foi possível alcançar este objetivo graças ao apoio e aparticipação dos professores, colegas, familiares e amigos que conviveram comigo nesteperíodo. Desta forma, gostaria de agradecer as pessoas e as instituições que contribuíram para arealização deste trabalho.

Meu agradecimento especial ao Prof. Dr. Ricardo Melo Bastos, que não hesitou em dar suas importantes contribuições ao longo de todo o processo, indicando novos caminhos e incentivando a continuidade do trabalho, além de contar com sua amizade e disponibilidade.

Ao Prof. Dr. Marcelo Blois Ribeiro, que acompanhou grande parte dos passos deste trabalho e demonstrou ser um bom conselheiro com suas questões desafiadoras e provocativas que engrandeceram esta tese.

Ao Programa de Pós-Graduação em Ciência da Computação da PUCRS, por proporcionar excelentes disciplinas que contribuíram para minha formação e servir de referência para meus ideais acadêmicos.

A Coordenação de Ensino do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, por viabilizar minha participação no doutorado.

Aos amigos que não vou enumerar por serem muitos, mas que sem eles eu não teria tido a alegria de concluir esta etapa de minha vida acadêmica.

Ao meu irmão, Fernando Covolan Rosito, pelo incentivo e apoio no meu trabalho, mostrando que a distância física não nos afastou como família.

Um agradecimento especial aos meus pais, Mário Antonello Rosito e Laura Maria Covolan Rosito, por terem entendido a importância deste doutoramentoe pelos valores éticos e morais que herdei.

Um agradecimento mais que especial a minha esposa Géssica Popow Rosito, pelas demonstrações de amor e carinho que recebi, por ter compartilhado todos os momentos e compreendido as dificuldades do caminho percorrido, e ter-me proporcionado nossa maior felicidade, o nosso querido filho Enzo Popow Rosito.

A todos vocês dirijo o meu sincero muito obrigado!

RECONFIGURAÇÃO DINÂMICA DE PROJETOS DE SOFTWARE: UM MODELO PARA INTEGRAR A GERÊNCIA DE PROJETOS DE SOFTWARE COM OS FLUXOS ORGANIZACIONAIS

RESUMO

Projetos de software são muito dinâmicos e requerem recorrentes ajustes de seus planos de projeto. Estes ajustes podem ser entendidos como reconfigurações na programação, alocação de recursos, entre outros elementos de *design*. A grande quantidade de informações que o gerente de projetos deve lidar, combinado com as frequentes mudanças no escopo e planejamento, torna essa atividade ainda mais desafiadora. Além disso, o gerente pode precisar consultar outros departamentos da organização durante o planejamento e a execução de um projeto de software. Com base nessas premissas, foi realizado um estudo para determinar o estado da arte sobre a reconfiguração dinâmica de projetos de software, com ênfase na integração da gerência de projetos e os fluxos organizacionais, a fim de identificar as principais lacunas e desafios de investigação nesta área. Para alcançar este objetivo, foi adotada uma metodologia de revisão sistemática. No entanto, conforme análise dos resultados, mesmo os trabalhos mais recentes não apresentaram uma solução que atenda todos os problemas da reconfiguração dinâmica ao mesmo tempo.

A fim de contribuir para a solução das dificuldades verificadas, esta tese apresenta um modelo computacional para a reconfiguração dinâmica de projetos de software em tempo de execução, com foco na integração das atividades específicas dos projetos com as atividades que fazem parte dos fluxos organizacionais.

Uma ferramenta de software foi desenvolvida para demonstrar e avaliar os resultados de um estudo experimental realizado com estudantes de pós-graduação em gestão de projetos. Além disso, este modelo considera a possibilidade de simular cenários de projetos de software, utilizando redes de Petri, em resposta a eventos de reconfiguração do projeto.

Palavras Chave: Reconfiguração Dinâmica, Fluxos Organizacionais, Gerência de Projetos de Software, Revisão Sistemática, Estudo Experimental.

DYNAMIC RECONFIGURATION OF SOFTWARE PROJECTS: A MODEL TO INTEGRATE SOFTWARE PROJECT MANAGEMENT WITH ORGANIZATIONAL WORKFLOWS

ABSTRACT

Software projects are very dynamic and require recurring adjustments of their project plans. These adjustments can be understood as reconfigurations in the schedule, in the resources allocation and other design elements. The large amount of information that the project manager must deal, combined with the frequent changes in the scope and planning, makes this activity more challenging. In addition, the manager may need to consult other departments in the organization during the execution of a software project. Based on these assumptions, a study was conducted to determine the state of the art of the dynamic reconfiguration of software projects, with emphasis on the integration of project management and organizational workflows, in order to identify the main gaps and challenges of research in this field. To reach this goal a methodology of systematic review was adopted. However, according to the analysis of the results, even the most recent studies did not present a solution that addresses all the problems of dynamic reconfiguration at the same time.

In order to contribute to the solution of the noted difficulties, this thesis presents a computational model for dynamic reconfiguration of software projects at runtime, with specific focus on the integration of project activities with activities that are part of the organizational workflows.

A software tool was developed to demonstrate and evaluate the results of an experimental study withpostgraduate students of project management. Moreover, this model considers the possibility to simulate software projects scenarios, using Petri nets, in response to events of project reconfiguration.

Keywords: Dynamic Reconfiguration, Organizational Workflows, Software Project Management, Systematic Review, Experimental Study.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O aumento do volume e da complexidade dos projetos que um gerente deve lidar ao mesmo tempo contribuem para os crescentes desafios relacionados ao desenvolvimento de projetos [KER00] [PRE09]. Particularmente em projetos de software, aspectos como as incertezas na especificação dos requisitos e sua instabilidade ao longo do desenvolvimento do projeto, no uso de tecnologia aplicada e da própria natureza humana, potencializam essas dificuldades.

O desenvolvimento de software requer o planejamento e a execução das atividades definidas de acordo com o escopo do projeto, no qual é necessário lidar tanto com questões técnicas quanto gerenciais. Em geral, as empresas estão organizadas de forma a gerenciar múltiplos projetos simultaneamente [LEE04]. Além disso, ao contrário do modelo tradicional de gerenciamento de projetos (que descreve os projetos individualmente), os modelos de gestão multiprojetos devem lidar com as interdependências entre os diferentes projetos com um conjunto de restrições relacionado à disponibilidade de tempo e de recursos [ADB91].

Os gestores também devem considerar que, durante o planejamento e execução de projetos de software, diferentes tipos de tarefas são atribuídos a recursos com características diferentes (resultando num conjunto complexo de dependências entre as atividades) a fim de alcançar as metas relacionadas ao tempo e aos custos desses projetos. Assim, em resposta às novas informações ou estimativas, os gestores podem precisar fazer alterações no plano de projeto, tais como a realocação de recursos ou cancelamento de tarefas [JOS05]. Esses ajustes, necessários para o projeto de acordo com as modificações que ocorrem durante seu ciclo de vida, dão origem ao termoreconfiguração dinâmica de projetos.

Pesquisas nesta área têm tradicionalmente abordado este problema de uma perspectiva estática, com foco em aspectos relacionados ao planejamento inicial dos projetos. Mais recentemente, este problema tem sido tratado numa perspectiva dinâmica, através do qual os projetos se adaptam durante a sua execução, fato que envolve a adição, exclusão e modificação de atividades e a realocação de recursos [CAL08]. Tais mudanças, geralmente, determinam o impacto sobre os custos e prazos previamente

estabelecidos para o projeto. Ainda, os gestores devem estar conscientes dos riscos inerentes aos projetos de software.

Esta tese de doutorado está inserida dentro do escopo do projeto de pesquisa intitulado “Reconfiguração Dinâmica em Projetos de Desenvolvimento de Software” desenvolvido pelo CePES (Centro de Pesquisa em Engenharia de Sistemas) da PUCRS. Neste sentido, esta pesquisa contribui no desenvolvimento do modelo de reconfiguração dinâmica proposto por Callegari [CAL10], considerando o uso da arquitetura multiagentes proposta por Schlösser [SCH10]. Resumidamente, Callegari [CAL10] propôs um modelo de referência para realizar replanejamentos durante a fase de execução de projetos de software. Nesse modelo, Callegari [CAL10] sugere que, a partir da ocorrência de um evento que possa alterar o fluxo de execução atual do projeto, seja realizada uma reconfiguração sobre as atividades do projeto. Em função disso, este modelo segue um ciclo para chamadas de propostas (*Call For Proposals* - CFPs), onde se verificam as tarefas que foram afetadas em função de um determinado evento (tal como, a saída de um recurso ou o acréscimo de tempo em uma tarefa). Em seguida, as CFPs são repassadas aos componentes do modelo denominados *solvers*, cuja função consiste em receber a chamada de propostas, analisá-las e gerar propostas para a realocação de recursos que permitam prosseguir a execução do projeto. Entretanto, esta proposta específica não aborda profundamente problemas relacionados ao sequenciamento das atividades de um projeto de software (o protótipo desenvolvido considera um cronograma fixo, ou seja, não há modificação na ordem de sequência das atividades). Ainda, considerando este modelo de reconfiguração dinâmica de projetos, Schlösser [SCH10] propôs um *solver* baseado em sistemas multiagentes (SMA). Esta última pesquisa apresenta uma arquitetura baseada em sistemas multiagentes, onde os agentes componentes da solução utilizam o protocolo de rede de contratos [SMI80] para coordenar o processo de geração de propostas.

Durante a execução dos projetos, o gerente pode ter que interagir com outros departamentos da organização a fim de obter informações relevantes para o projeto (tal como, contatar o departamento de recursos humanos sobre a necessidade de contratar mais profissionais para o projeto). Desta forma, pode-se observar que existe uma distinção entre as atividades específicas em um projeto e as atividades que fazem parte do fluxo comum de trabalho da organização (aqui chamado de **fluxo organizacional**). Sendo assim, o gerente de projetos deve lidar com essa dissociação entre o fluxo de

atividades de um projeto de software e os fluxos organizacionais (este último procura oferecer algum tipo de suporte para o projeto de software). Ambos os tipos de fluxos de trabalho são executados em paralelo, têm seus próprios recursos e podem influenciar no tempo das atividades e nos custos do projeto. O gerente de projeto pode precisar, portanto, de algum tipo de suporte durante o processo de tomada de decisão, levando em conta a integração dos diferentes conjuntos de atividades durante a execução simultânea de projetos de software.

Esta pesquisa considera o uso de redes de Petri (RdP) para simular o comportamento dinâmico dos projetos de software. A rede de Petri é uma ferramenta gráfica e matemática para a modelagem, análise e verificação de sistemas [MUR89]. Assim, as redes de Petri podem ser aplicadas em diversas áreas, tais como a modelagem de sistemas de banco de dados distribuídos, projeto e análise de *workflows* e processos de negócios, entre muitos outros [MU89] [ZHO95]. A RdPé considerada uma importante ferramenta de análise devido asua capacidade de ser descrita como sendo um conjunto de equações algébricas [MU89] [ZHO95] [VAN98]. Dessa forma, as redes de Petri podem ser utilizadas para auxiliar no projeto, análise e simulação dos vários cenários que ocorrem em projetos de software.

Com base nessas premissas, foi realizado um estudo para determinar o estado da arte sobre a reconfiguração dinâmica de projetos de software, com ênfase na integração da gestão de projetos com os fluxos organizacionais, a fim de identificar as principais lacunas e desafios de investigação neste campo. Para alcançar este objetivo, foi adotadaa metodologia de revisão sistemática. A revisão sistemática é uma prática de pesquisa, utilizada muitas vezes na área médica, que foi adaptada para a engenharia de software [BIO05] [KIT04]. Este processo ajuda a estabelecer um rigor científico necessário para definir o estado da arte e para produzir resultados mais confiáveis. Os resultados deste trabalho mostram que existe uma diversidade de técnicas e estratégias para a reconfiguração dinâmica de projetos de software. No entanto, os resultados obtidos revelaram que nenhuma pesquisa considera a integração das atividades dos projetos de software com as atividades que fazem parte de fluxo de trabalho comum da organização durante a reconfiguração dos projetos em tempo real.Neste sentido, esta pesquisa propõe um modelo computacional, chamado de*Software Planning Integrated Model* (SPIM), para fornecer o suporte necessário para a reconfiguração dinâmica de projetos de software em

ambientes multiprojetos, considerando a integração destes projetos com os fluxos organizacionais.

A fim de ilustrar a operacionalidade dos conceitos propostos pelo modelo SPIM e o seu conjunto de regras, foi desenvolvido um software chamado *Software Project Integrated Tool* (SPIT). O SPIT atua como um *add-in* para uma ferramenta comercial de gerenciamento de projetos muito conhecida e utilizada nas empresas de software.

As seções seguintes apresentam um detalhamento sobre a motivação, seus objetivos e asatividades desta pesquisa.

1.1 Motivação da Tese

Projetos de software frequentemente experimentam diferentes alterações durante seus ciclos de vida [SCH02]. Dois exemplos comuns são as frequentes realocações de recursos e o replanejamento de atividades. Este cenário se torna ainda mais desafiador quando recursos e atividades são compartilhados por mais de um projeto. Como resultado, o gerente deve revisar constantemente o plano do projeto, ajustando o cronograma e a alocação dessas atividades, a fim de trazer o projeto para um plano exequível.

No entanto, deve ser considerada a integração destas atividades específicas dos projetos com as atividades que fazem parte do fluxo de atividades comum da organização. O gerente de projetos, por exemplo, pode entrar em contato com o departamento financeiro sobre a necessidade de aquisição de algum equipamento para um determinado projeto de software (estas atividades fazem parte de um fluxo organizacional desta empresa). As atividades do projeto de software,entretanto, podem depender da execução dos fluxos organizacionais (resultando em uma relação de dependência entre estes dois fluxos diferentes). Pode ser observado, portanto, que os gestores precisam de algum tipo de apoio para lidar com esta situação.

Em um trabalho recente [ROS12], verificou-se através de uma revisão sistemática que existe uma diversidade de técnicas e estratégias para reconfiguração dinâmica de projetos de software. A revisão sistemática é uma metodologia de pesquisa que usa métodos sistemáticos para identificar, selecionar e avaliar criticamente os estudos científicos em um campo específico de investigação. É uma revisão planejada para responder a uma pergunta específica que pode ou não incluir métodos estatísticos. Métodos estatísticos utilizados na análise e síntese dos estudos selecionados são

chamados de meta-análise [HIG08]. Como resultado desta pesquisa, observou-se que poucas abordagens consideram a integração das atividades dos projetos de software com as atividades que fazem parte de fluxo de trabalho comum da organização. Entretanto, nenhuma abordagem apresentou uma solução prática para este problema.

Conforme mencionado anteriormente, o gerente de projetos precisa lidar tanto com assuntos gerenciais quanto técnicos durante o planejamento e a execução das atividades, levando em consideração não só as atividades do projeto de software, mas também as atividades que pertencem de forma compartilhada aos demais fluxos de atividades de suporte aos projetos da empresa (os fluxos organizacionais). Esta solução de reconfiguração dinâmica de projetos de software, consequentemente, deve considerar a complexidade de identificar a interdependência entre as atividades dos fluxos de trabalho da organização e o fluxo de trabalho do projeto.

Deve-se considerar, ainda, que a maioria dos projetos nas empresas de software é realizada em ambiente multiprojetos. Isso significa que os projetos não estão isolados uns dos outros, estes estão ligados através de determinadas relações, tais como: a competição e partilha de recursos, conflito de agendas, conflito entre os agentes, etc. Devido a estas relações mútuas, especialmente o conflito de recursos em vários projetos simultâneos, o gerenciamento de projetos torna-se cada vez mais complexo.Observa-se, desta forma, aimportância daspesquisas relacionadas à resoluçãodo problema da programação de múltiplos projetos com recursos limitados (RCMPSP), que é definido como um problema de otimização combinatória que lida com dois tipos de restrições: de precedência de atividades e de recursos. Do ponto de vista de complexidade, este problema pertence à classe *NP-hard*.

Em vista deste cenário, uma série de pesquisas sobre RCMPSP têm sido relatadas na literatura [FRI00].Estes trabalhos têm lidado com diferentes variedades de situações, geralmente, com base em projeto único,onde um ou ambos os tipos de restrições são simplificados.Fricke e Shenhar [FRI00] consideraram que os fatores de sucesso mais importantespara os ambientesmultiprojetos diferem dos fatores de sucesso da gestão tradicional de um único projeto, e estes foram consistentes com outras pesquisas emergentes relacionadas ao ambiente de desenvolvimento de produtos. Lova e Tormos [LOV01] analisaram o efeito dos esquemas de geração da programação de atividades, tais como a execução em série ou em paralelo, as regras de priorização de atividades e a definição *first-come first-served* em ambientes de um único projeto e de vários

projetos.Liao et al. [LIA02] apresentaram um algoritmo heurístico para resolver problema da programação de múltiplos projetos com recursos limitados e discutiram a sua aplicação em uma fábrica. Kim et al.[KIM05] propuseram um algoritmo genético híbrido com controlador de lógica *fuzzy* para resolver o RCMPSP. A abordagem proposta foi baseada no projeto de operadores genéticos com controlador de lógica *fuzzy*.

A revisão bibliográfica realizada nesta pesquisa revelou quea maioria das soluções propostas ainda se baseia em modelos de gerenciamento de projetos convencionais, como o método do caminho crítico (CPM) ou a técnica de avaliação e revisão de programas (PERT). Embora o método CPM/PERT seja aplicado com sucesso na gestão de um único projeto, este apresenta dificuldades na gestão de vários projetos simultâneos, uma vez que este modelo não considera a restrição de recursos.Além disso, uma série de fatores como a interdependência, criticidade, prioridades conflitantes e a substituição de recursos, que afetam o controle sobre a execução dos projetos, não podem ser representados pelo modelo tradicional CPM/PERT.Outros modelos de gestão, tais como o *venture ERT* (VERT), *graphical ERT* (GERT), as redes de atividades e as abordagens de matriz de influências, também não são capazes de incorporar esses fatores [RED01]. As redes de Petri, entretanto, podem incorporar esses fatores e apresentar uma metodologia gráfica e analítica poderosa na gestão de projetos.

A rede de Petri é uma ferramenta de modelagem gráfica e matemática que tem sido aplicada com sucesso nas áreas de sistemas dinâmicos a eventos discretos (DEDS), tais como na avaliação de desempenho, protocolos de comunicaçãoe modelos de decisão [YUA05]. A principal diferença entre as redes de Petri e as outras ferramentas de modelagem é a presença de fichas (*tokens*), que são utilizadas para simular as atividades simultâneas, dinâmicas, e assíncronas em um sistema. O comportamento de um sistema pode ser representado em uma rede de Petri através da criação de equações de estado (*state equations*), equações algébricas e outros modelos matemáticos. A gestão de projetos, desta forma,é considerada como uma área potencial para o uso da modelagem com redes de Petri [PET81] [KIM95] [KUM98].

Esta tese de doutorado parte de um projeto de pesquisa desenvolvido junto a um grupo de professores e alunos participantes do CePES da PUCRS. Entre os projetos desenvolvidos neste centro, Callegari [CAL10], em sua tese de doutorado, apresenta um modelo de referência para tratar de eventos que podem alterar o planejamento de projetos de desenvolvimento de software em tempo de execução, em um contexto

multiprojetos onde recursos podem estar compartilhados. Este modelo de referência apoia-se em quatro subáreas para dar suporte à reconfiguração de projetos, sendo elas: (i) Seleção de recursos para participação de projetos de software a partir de um *pool* compartilhado; (ii) Alocação de recursos às tarefas componentes de projetos de desenvolvimento de software; (iii) Planejamento e replanejamento de tarefas e recursos associados e (iv) Integração dos projetos com fluxos organizacionais da empresa. Schlösser [SCH10], em sua dissertação de mestrado, contribui para o modelo de reconfiguração dinâmica proposto através de uma arquitetura baseada em sistemas multiagentes para dar suporte ao gerenciamento de agendas dos recursos que desenvolvem os projetos de software.

A pesquisa aqui apresentada contribui especificamente com a terceira e quarta área do modelo de referência proposto na tese de Callegari [CAL10], ou seja, no planejamento e replanejamento de tarefas e recursos associados; e na integração dos projetos com fluxos organizacionais da empresa. A presente pesquisa considera, também, o ambiente multiagentes proposto por Schlösser [SCH10].

1.2 Objetivo Geral da Pesquisa

O objetivo geral desta pesquisa é desenvolver e avaliar um modelo computacional para a reconfiguração dinâmica de projetos de software em tempo de execução, com foco na integração das atividades específicas dos projetos com as atividades que fazem parte dos fluxos organizacionais.

1.2.1 Objetivos específicos

O objetivo geral pode ser desmembrado nos seguintes objetivos específicos:

- Identificar as características fundamentais para uma solução computacional de reconfiguração dinâmica de projetos de software, que envolva o uso de recursos compartilhados, a integração com os fluxos organizacionais e a execução em um ambiente de múltiplos projetos simultâneos;
- Elaborar um modelo que descreva a relação entre os diversos elementos advindos da gestão de projetos e dos processos de desenvolvimento de software. Com relação à visão sobre o gerenciamento dos projetos, optou-se por usar como referência o Guia do PMBOK [PMI08]. Para os processos de desenvolvimento de software, foram pesquisados o RUP [KRU00], OPEN [FIR02] e SPEM [OMG12];

- Desenvolver uma estratégia que permita a reconfiguração da programação das atividades de projetos de software, considerando a integração das atividades específicas dos projetos com as atividades que fazem parte dos fluxos organizacionais;
- Implementar um protótipo de ferramenta que possibilite avaliar os conceitos propostos nesta pesquisa;
- Avaliar a proposta de soluçãopor meio do protótipo desenvolvido, utilizando a simulação de cenários com redes de Petri.

Uma vez apresentado os objetivos geral e específicos deste estudo, introduz-se a questão de pesquisa utilizada:

Questão da Pesquisa

"Como realizar a reconfiguração dinâmica de projetos de software, em tempo de execução, no que concerne à programação das atividades, considerando a integração do fluxo de trabalho de um projeto de software com os demais fluxos de trabalho da organização?"

1.3 Metodologia de Pesquisa

Embora alguns trabalhos relacionados com o objetivo desta pesquisa tenham sido encontrados na revisão teórica desenvolvida, não se tem conhecimento de que o problema apresentado tenha sido tratado da mesma maneira em outros estudos. Desta forma, esta pesquisa se caracteriza como exploratória, sendo o principal método de pesquisa utilizado, de acordo com a classificação de Oates [OAT06], projeto e criação (do inglês *design and creation*).

Segundo Oates [OAT06], a estratégia de pesquisa chamada projeto e criação têm como finalidade o desenvolvimento de novos produtos de tecnologia da informação (TI) que podem ser construções, modelos, métodos ou instanciações. Para este último caso, considera-se como instanciação um trabalho que demonstra que modelos, métodos, gêneros ou teorias podem ser implementados em um sistema de computador. Salienta-se que, para que os projetos que seguem a estratégia de projeto e criação possam ser considerados de fato uma pesquisa, estes devem demonstrarqualidades acadêmicas como análise, discussão, justificativa e avaliação crítica [OAT06].

Nesta pesquisa, os novos produtos de TI desenvolvidos envolvem um modelo computacional e um protótipo de ferramenta. As avaliações do modelo foram realizadas com o auxílio do protótipo de ferramenta desenvolvido. A primeira avaliaçãofoi realizada através de uma comparação, em um estudo experimental, entre a proposta de planejamento integrado e a proposta tradicional de planejamento de projetos de software. A segunda avaliação foi realizada de forma analítica, ou seja, verificando-se os resultados produzidos pelo modelo em função da ocorrência de eventos sobre cenários simulados.

A seção a seguir fornece detalhes sobre a metodologia e a geração do modelo de reconfiguração dinâmica de projetos de software.

1.3.1 Atividades da Pesquisa

A elaboração do modelo de reconfiguração dinâmica proposto nesta pesquisa passou por um conjunto de etapas preliminares. Inicialmente, para obter maior conhecimento sobre o assunto de pesquisa, realizou-se um levantamento bibliográfico e estudo do referencial teórico que permitiu aprofundar os conhecimentos sobre processos de desenvolvimento e sobre o gerenciamento dos projetos de software.

O estudo sobre processos de desenvolvimento envolveu o Processo Unificado, por sua absorção na indústria de software, o *OPEN Process Framework*, devido o seu destaque na comunidade acadêmica, e o SPEM 2.0 (*Software & Systems Process EngineeringMeta-Model Specification*), que é um metamodelo desenvolvido pelo *Object Management Group* para a definição de processos de desenvolvimento de software e seus componentes. Esses modelos foram escolhidos porque possuem um bom grau de detalhamento em termos de documentação, inclusive apresentando seus respectivos metamodelos [KRU00] [GRA97] [OMG12]. Com relação à visão sobre o gerenciamento dos projetos, optou-se por usar como referência o Guia do PMBOK (*Project Management Body of Knowledge*) [PMI08]. Estes modelos encontram-se no Capítulo 2.

Em seguida, observou-se a necessidade de integração das duas visões (gerencial e técnica) para servir de base aos estudos seguintes. Desta forma, elaboraram-se modelosque proporcionaram a integração das duas visões, considerando os diferentes processos de desenvolvimento de software estudados. Estes modelos de integração encontram-se no Capítulo 3.

Depois da definição desses modelos de base, foi proposto um modelo de integração entre a gerência de projetos e os processos de desenvolvimento de

software,que auxilia o planejamento integrado das atividades relacionadas a estas duas áreas de conhecimento. Nesta etapa, também se desenvolveu um protótipo que possibilita o uso do modelo proposto. Os resultados dessa integração já foram explorados em artigos publicados [ROS06] [CAL08] [ROS08a] [ROS08b] [ROS12b].

Após o desenvolvimento desta primeira versão do protótipo, foi realizado dois estudos experimentais. O primeiro experimento foi realizado com alunos de graduação e pós-graduação da área da Ciência da Computação. Após, novas funcionalidades foram implementadas no protótipo, e um segundo estudo experimental foi realizado com trinta e seis alunos de cursos de pós-graduação em Gerenciamento de Projetos de duas instituições de ensino superior distintas. Os resultados destes experimentos foram publicados em artigos[ROS13] [ROS14]. Ainda nesta etapa, foi realizada a consolidação e a avaliação dos resultados obtidos.

Em seguida, iniciou-se um estudo aprofundado sobre o tema principal desta pesquisa. Desta forma, realizou-se uma revisão sistemática para identificar os requisitos necessários para o desenvolvimento de uma solução computacional que permita a reconfiguração dinâmica da programação de projetos de software, considerando múltiplos projetos simultâneos e a sua integração com os demais fluxos organizacionais. Os resultados desta revisão sistemática foram publicados em um artigo [ROS12a]. Ainda, objetivou-se ampliar o estudo sobre a formalização e definição dos tipos de eventos que podem afetar uma configuração de projeto de software.

Desse estudo, produziu-se um modelo computacional sobre a reconfiguração dinâmica de projetos de software. Desta forma, objetivou-se desenvolver uma estratégia para a reconfiguração dinâmica de projetos de software de maneira a ajustar o sequenciamento das atividades dos projetos, considerando um conjunto de restrições sobre as atividades, os recursos e as informações sobre as variáveis globais de projeto. Os detalhes sobre esta arquitetura foram publicados em [ROS12c]. A ênfase da reconfiguração tratada nessa pesquisa, desta forma, ocorre sobre as atividades (impactando também na alocação de recursos humanos para tais atividades) em um contexto de múltiplos projetos de software que são executados de forma concorrente. Ainda, procurou-se desenvolver uma solução que permitisse a integração clara e objetiva dos projetos de software com os fluxos organizacionais.

Depois da definição deste modelo computacional de reconfiguração dinâmica, desenvolveu-se uma nova versão do protótipo para oferecer suporte automatizado para o

planejamento e replanejamento das atividades de projetos de software. Ainda nesta fase, foi realizada a consolidação e avaliação dos resultados obtidos, permitindo a definição das conclusões desta pesquisa. A Figura 1 resume as principais atividades da pesquisa.

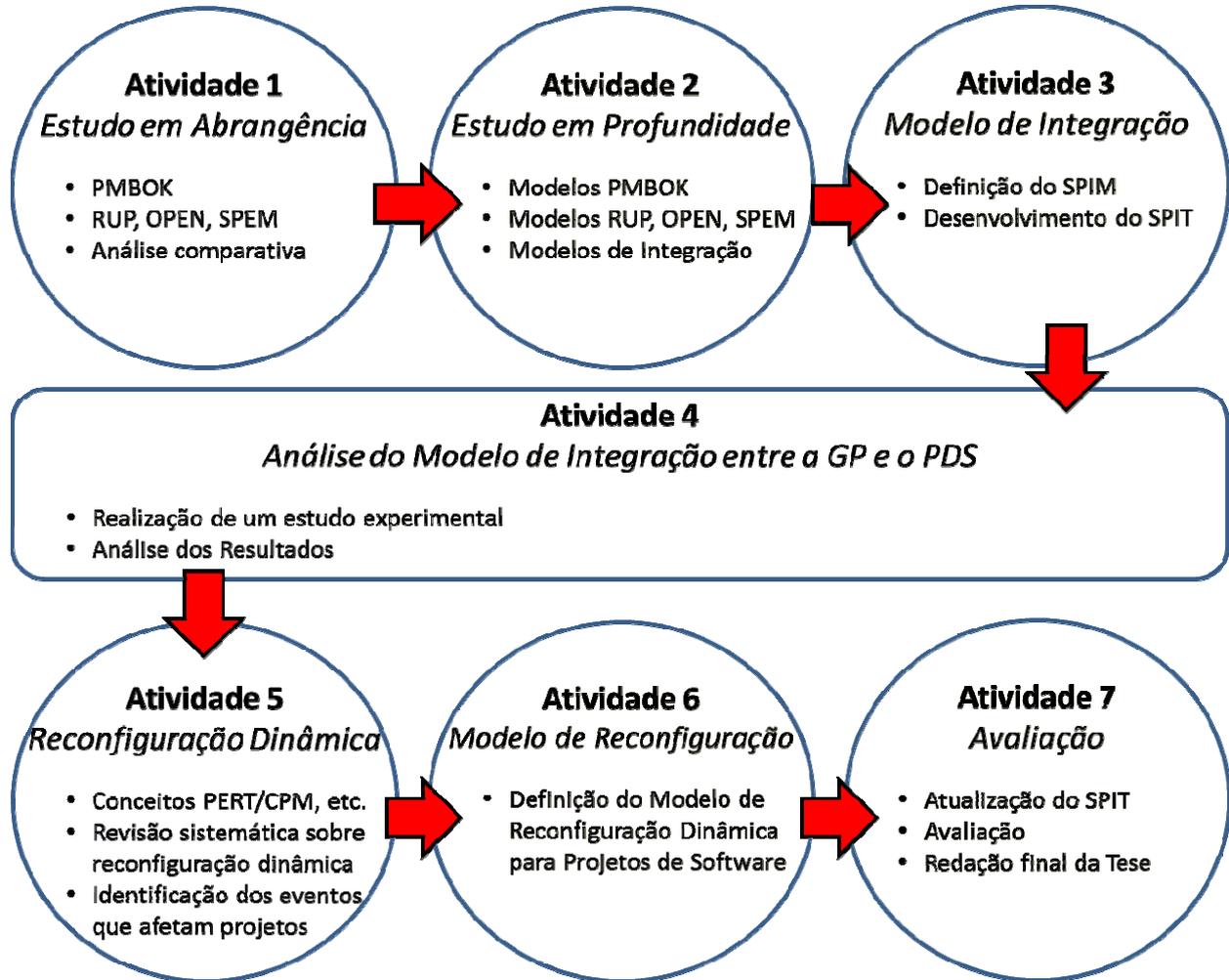



Figura 1. Atividades da pesquisa

A seção seguinte explica como o texto desta tese de doutorado está organizado.

1.4 Organização do Texto

O texto desta tese está estruturado da seguinte forma: o Capítulo 2 apresenta uma caracterização da área de estudo, destacando os termos e definições utilizados no decorrer do texto.

O Capítulo 3 comenta sobre os modelos de referência que foram usados como base para a elaboração do modelo de reconfiguração. Os modelos de referência foram comparados e combinados em um modelo integrado. Um estudo sobre a reconfiguração dinâmica de projetos, resultado de uma revisão sistemática,é apresentado no Capítulo 4,

enquanto que o Capítulo 5 apresenta o estudo realizado sobre a utilização de redes de Petri na gestão de projetos.

O Capítulo 6 apresenta o modelo de referência elaborado para reconfiguração dinâmica propriamente dito, além do protótipo desenvolvido. A avaliação do modelo de reconfiguração é comentada no Capítulo 7, onde também são detalhados os cenários usados para a geração dos dados e dos comportamentos obtidos como resposta. Conclusões e demais indicações de trabalhos futuros encontram-se no Capítulo 8.

2 PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE E A GESTÃO DE PROJETOS

Para um bom entendimento do modelo proposto e dos trabalhos relacionados faz-se necessário a compreensão dos fundamentos teóricos e metodologias básicas nos quais esta pesquisa se baseia. Desta forma, este capítulo apresenta uma introdução sobre os processos de desenvolvimento de software, a gestão de projetos e sobre outros conceitos utilizados no contexto desta pesquisa.

2.1 Processo de Desenvolvimento de Software: Definição e Aspectos Relacionados

De acordo com Jacobson, Booch e Rumbaugh [JAC01], um processo de desenvolvimento de software é o conjunto completo de atividades necessárias para transformar os requisitos dos usuários em produtos de software. Como as organizações geralmente consideram o desenvolvimento de um produto de software como um projeto, então um processo pode ser considerado como uma sequência de passos que um projeto pode seguir para desempenhar alguma tarefa.

Para Sommerville [SOM06], um processo de desenvolvimento de software é um conjunto de atividades e resultados associados que levam a produção de um produto de software. Ainda, segundo Fuggetta [FUG00], o processo de desenvolvimento de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para compreender, desenvolver e manter um produto de software.

Definir um processo de desenvolvimento de software envolve várias informações como atividades a serem desempenhadas, recursos utilizados, artefatos consumidos e gerados, dentre outras [FAL98] [WER96] apud [MAC00]. Para Derniame, Kaba e Warboys [DER99], os principais conceitos ligados à sua modelagem são:

- Atividade: operação atômica ou composta, ou uma etapa de um processo. As atividades visam gerar ou modificar um conjunto de artefatos, incorporando e executando procedimentos, regras e políticas organizacionais. Algumas atividades podem ser decompostas em subatividades, embora isso não seja obrigatório;

- Artefato: informação desenvolvida e mantida em um projeto de software. Alguns artefatos podem ser decompostos em subartefatos, embora isso não seja obrigatório;
- Direção: são procedimentos, regras e políticas organizacionais que dirigem atividades e geralmente estão estruturados na forma de manuais;
- Recurso: é um fator necessário na execução de uma atividade. Os recursos devem ser divididos em: executores (agentes humanos do processo) e ferramentas (agentes computadorizados que são usados tradicionalmente no desenvolvimento de software).

Outra informação importante ligada à definição de processos de software é o que se deseja alcançar a partir de sua utilização. Nesse sentido, Tyrrell [TYR00] define um conjunto de objetivos que os processos de software devem atender:

- Efetividade: processos de desenvolvimento de software devem ajudar a determinar as necessidades do cliente e verificar se o que foi produzido satisfaz o cliente.
- Manutenibilidade: o processo de desenvolvimento de software deve ser capaz de expor a maneira de pensar de projetistas e programadores de forma que suas intenções sejam claras. Assim, torna-se fácil encontrar e reparar falhas no produto.
- Previsibilidade: o processo de desenvolvimento de software deve ajudar a predizer quanto tempo será necessário para o desenvolvimento de cada parte do produto.
- Repetível: produzir um processo novo para cada projeto implica em grandes gastos para a organização. Dessa forma, é importante que o processo de desenvolvimento de software seja criado de forma a poder ser reutilizado em vários projetos.
- Qualidade: o processo de desenvolvimento de software deve permitir engenheiros de software assegurar um produto de qualidade elevada. Para isso, o processo deve fornecer uma ligação entre os desejos de um cliente e o produto de um desenvolvedor.
- Melhoria: o processo de desenvolvimento de software deve ser constantemente melhorado. Dessa forma, o próprio processo deve permitir a identificação de pontos de melhoria.

- Rastreabilidade: o processo de desenvolvimento de software deve permitir que a gerência, os desenvolvedores e o cliente sigam o status do projeto.

Além dos conceitos e objetivos acima, é importante, também, que os processos de desenvolvimento de software sejam padronizados de forma a garantir uma maior qualidade dos produtos de software [MAC00].

2.2 Processo Padrão de Desenvolvimento de Software

Um processo padrão de desenvolvimento de software pode ser definido com sendo o processo que deve servir como referência para guiar a execução de todos os projetos de desenvolvimento dentro de uma organização. Segundo Ginsberg e Quinn [GIN95], o processo padrão é o meio pelo qual a organização expressa os requisitos que todos os processos de desenvolvimento de software dos projetos devem atender. Conforme Humphrey [HUM89] apud [COE03], e Xu e Runeson [XU05], as vantagens da implantação de um processo padrão de desenvolvimento de software são as seguintes:

- Redução dos problemas relacionados a treinamento, revisões e suporte de ferramentas;
- Experiências adquiridas em cada projeto podem ser incorporadas ao processo padrão, contribuindo para sua melhoria;
- Maior facilidade em medições de processo e qualidade;
- Maior facilidade de comunicação entre os membros da equipe;
- Melhor adequação de novos membros na equipe de projeto;
- Melhor desempenho, previsibilidade e confiabilidade dos processos de trabalho.

Atualmente, devido a importância da utilização de um processo padrão, um esforço considerável tem sido devotado para sua elaboração e, como forma de auxiliar a sua criação, foram desenvolvidos diversos processos tais como: *Rational Unified Process* (RUP) [KRU00], *Extreme Programming* (XP) [BEC99] e *Object-oriented Process, Environment and Notation* (OPEN) [OPE09]. Eles podem ser adotados por completo, ou permitem ainda ser adaptados de acordo com as características da organização. É possível também que uma organização se baseie em mais de um desses processos para a definição de seu processo padrão.

Entretanto, independente da estratégia escolhida para a adoção do processo padrão de desenvolvimento de software, é importante considerar que seu uso é de suma

importância, já que ele é um dos mais importantes mecanismos para gerenciar e controlar projetos e produtos de software [HUM91].

2.3 Gestão de Projetos de Software

Antes de examinar a gerência de projetos, entretanto, é importante compreender o conceito de projeto. A definição sobre o termo projeto utilizada nesta pesquisa reúne os conceitos proporcionados por Kerzner [KER00], *Project Management Institute*[PMI08] e Schwalbe [SCH02]. Com base nessas fontes, compreende-se que um projeto é um empreendimento temporário com objetivo único, que consome recursos tais como pessoas, verbas e materiais necessários, além de ser desenvolvido no ambiente de organizações sob pressões de prazo, custos e qualidade.

Segundo estudos empíricos, a eficácia de uma organização depende, em boa parte, do sucesso de seus projetos [KER00]. Desta forma, muitos pesquisadores trabalham na investigação dos fatores de sucesso dos projetos, tais como a definição de produto, qualidade de execução e técnicas de gerência de projetos [PRE09]. Observa-se, entretanto, que alguns fatores são determinantes para o sucesso do projeto, tais como: metas claramente definidas; competência da gerência de projetos; adequada alocação de recursos; comunicação eficiente; planejamento, programação e controle eficientes; capacidade de atualização de dados e informações; apropriação de técnicas e tecnologias [PRE09] [SOM06] [SCH02].

Os projetos de desenvolvimento de software, escopo desta pesquisa, possuem peculiaridades devido a própria complexidade inerente ao software. Dentre essas peculiaridades citadas por Bezerra [BEZ07], Plekhanova [PLE98], Schwalbe [SCH02], Dong et al. [DON08] e Jurison [JUR99], destacam-se a volatilidade dos requisitos (fato que incide nas constantes alterações nas especificações do produto), a necessidade de pessoas especializadas para o desenvolvimento das tarefas, e a intangibilidade (o que torna difícil medir o progresso de projetos de software). Complementar a estas peculiaridades, observa-se que eventos não previstos podem ocorrer durante a execução dos projetos, provocando a necessidade de reexaminar e atualizar o seu planejamento [DEB07] [SOD08].

Para lidar com os obstáculos da área de software, de forma a alcançar com sucesso os objetivos dos projetos, faz-se necessário gerenciá-los. A gerência de projetos, segundo Schwalbe [SCH02], é responsável pelo controle da realização dos objetivos do

projeto através da aplicação de um conjunto de técnicas e ferramentas. A gestão, conforme o *Standish Group International* [SGI10], deve abranger o ciclo de vida completo de um projeto, ou seja, passando pelas fases de: planejamento, execução e controle. Para o *Project Management Institute* [PMI08], um projeto é concluído com sucesso quando as especificações de escopo, tempo e custo estão conforme o planejado.Adicionalmente, as organizações que desenvolvem software estão inseridas em ambientes com multiprojetos, onde as dificuldades encontradas em um único projeto intensificam-se [ARA09].

Observa-se, desta forma, que as dificuldades encontradas no gerenciamento de projetos de desenvolvimento de software estão diretamente relacionadas às características intrínsecas do software. Para o gerenciamento de projetos de software, consequentemente, a organização precisa de um modelo que descreva a complexidade do projeto, envolvendo recursos, tempo, custos e objetivos. Entretanto, a maioria dos modelos ou guias voltados para a gerência de projetos, tal como o *Project Management Body of Knowledge Guide* (PMBOK), não se dirigem especificamente a processos de desenvolvimento de software. Além disso, os processos de desenvolvimento de software, por sua vez, geralmente fornecem apenas um conjunto de práticas que atendem a determinadas atividades e fluxos de trabalho relacionados à gerência de projetos [HEN00] [ROS08a].

2.4 Metamodelos e Modelos de Gestão de Projetos e de Processos de Desenvolvimento de Software

Um metamodelo é um modelo que permite modelar outro modelo conceitual, ou seja, é uma forma de descrever como um modelo deve ser modelado [LEE02]. Os processos de software construídos a partir de um metamodelo, geralmente, oferecem um alto grau de formalismo e melhor suporte para consistência e customização, uma vez que os conceitos que formam sua base são explicitamente definidos.

Segundo Perez e Sellers [PER05], essa é a maneira de formalizar as ideias e conceitos subjacentes de um processo de software que são importantes para sua checagem de consistência e, também, para possíveis extensões ou modificações do processo.Além disso, os metamodelos são normalmente representados com diagramas de classe da *Unified Modeling Language* – UML [OMG10], de modo que eles são capazes de representar restrições através das multiplicidades entre suas metaclasses.Os

metamodelos também permitem associar restrições mais complexas através da linguagem natural ou linguagens formais, tais como, *Object Constraint Language* – OCL [WAR03], Z [MOU01], Lógica de Primeira Ordem (em inglês,*First-order Logic* – FoL) [SMU95], entre outras.A utilização de metamodelos também possibilita o suporte automatizado para os processos de software, o que é considerado de alta importância devido a quantidade e complexidade de informações envolvidas em um processo de software.

Existem diversos metamodelos de processo, tais como: *Software & Systems Process Engineering Meta-Model Specification - SPEM 1.1* [OMG02], *OPEN Process Framework* – OPF [FIR02], entre outros. Atualmente, o metamodelo referência para a definição de processos de software é o SPEM 2.0, que foi publicado pela *Object Management Group*(OMG) em 2007 [OMG12].Segundo a especificação do SPEM 2.0, a meta em definir um metamodelo de referência é viabilizar a criação de grande variedade de processos de software para diferentes culturas que utilizam diferentes níveis de formalismo e modelos de ciclo de vida.

Nas seções seguintes são apresentados os modelos utilizados como referência nesta pesquisa.

2.4.1 Project Management Body of Knowledge - PMBOK

2.4.1.1 Definição e Aspectos Relacionados

Reconhecido internacionalmente pelo seu esforço em definir normas e dar suporte aos profissionais de gerência de projetos, o *Project Management Institute* (PMI) [PMI08] publicou um guia geral de gerência de projetos, o *PMBOK Guide*. O *Project Management Body of Knowledge* reúne as melhores práticas aplicáveis à maioria dos projetos e sobre as quais há um amplo consenso sobre o seu valor e utilidade.

De acordo com o *Project Management Institute* [PMI08], o principal objetivo do *PMBOK Guide* é identificar um subconjunto dos conhecimentos sobre gerência de projetos que seja reconhecido, genericamente, como sendo uma coleção de boas práticas. Este documento define nove áreas de conhecimento da gerência de projetos: integração, escopo, custo, qualidade, recursos humanos, comunicações, riscos, aquisições e tempo. Destaca-se no presente trabalho a gerência do tempo do projeto, que é responsável por descrever os processos necessários para assegurar que o projeto termine dentro do prazo previsto. Os processos principais da gerência do tempo são os

seguintes: definição das atividades, sequenciamento das atividades, estimativa da duração das atividades, desenvolvimento do cronograma, e controle do cronograma. Cada um destes processos é caracterizado por entradas, ferramentas e técnicas, e saídas.

O PMBOK, porém, não é um processo em seu sentido estrito, pois não determina quais são as ações e nem indica como estas devem ser executadas para o correto desenvolvimento de um projeto. Além disso, o PMBOK é dito mais compatível com atividades industriais e, portanto, não aborda especificamente processos de desenvolvimento de software [SCH02] [CAL07].

2.4.1.2 Modelo de Referência para o PMBOK

Para efeito da discussão proposta por esta pesquisa, observa-se que os conceitos do PMBOK são, em sua maioria, representados por textos descritivos. Entretanto, objetivando comparar dois modelos e, posteriormente, executar a integração entre eles, deve-se representá-los sob estruturas compatíveis. Dessa forma, foi utilizado o metamodelo previamente desenvolvido em [CAL07] que utiliza a notação da linguagem UML para representar seus elementos.

Como podemos observar na Figura 2, este modelo foi desenvolvido observando os principais conceitos contidos no PMBOK, tais como programa, projeto, recursos, atividades, papéis, produtos e classes associadas. É importante salientar que este modelo foi desenvolvido com uma visão para atender as necessidades exclusivas dos projetos de desenvolvimento de software.

O PMBOK agrupa conhecimentos relacionados a comprovadas e amplamente aplicadas práticas tradicionais de gestão de projetos. Um dos principais conceitos de gestão de projetos é que as partes interessadas (classe `stakeholders`), que correspondem a todos os indivíduos e organizações que têm qualquer tipo de relação com um projeto [SCH02].

As partes interessadas participam das atividades (classe `Activity`) e tarefas (classe `Task`) que são classificadas em um número de áreas de conhecimento (classe `KnowledgeArea`), e que podem ter o apoio de ferramentas (classe `Tool`) e técnicas (classe `Technique`). Atividades consomem e produzem produtos de trabalho (classe `Deliverable`), utilizando os recursos apropriados. Além disso, a classe

`DeliverableType`define o tipo de produto de trabalho, tais como um contrato, uma proposta comercial e assim por diante. Mais do que apenas um conjunto de atividades, um projeto é definido como sendo um esforço temporário empreendido para criar um produto único, serviço ou resultado.

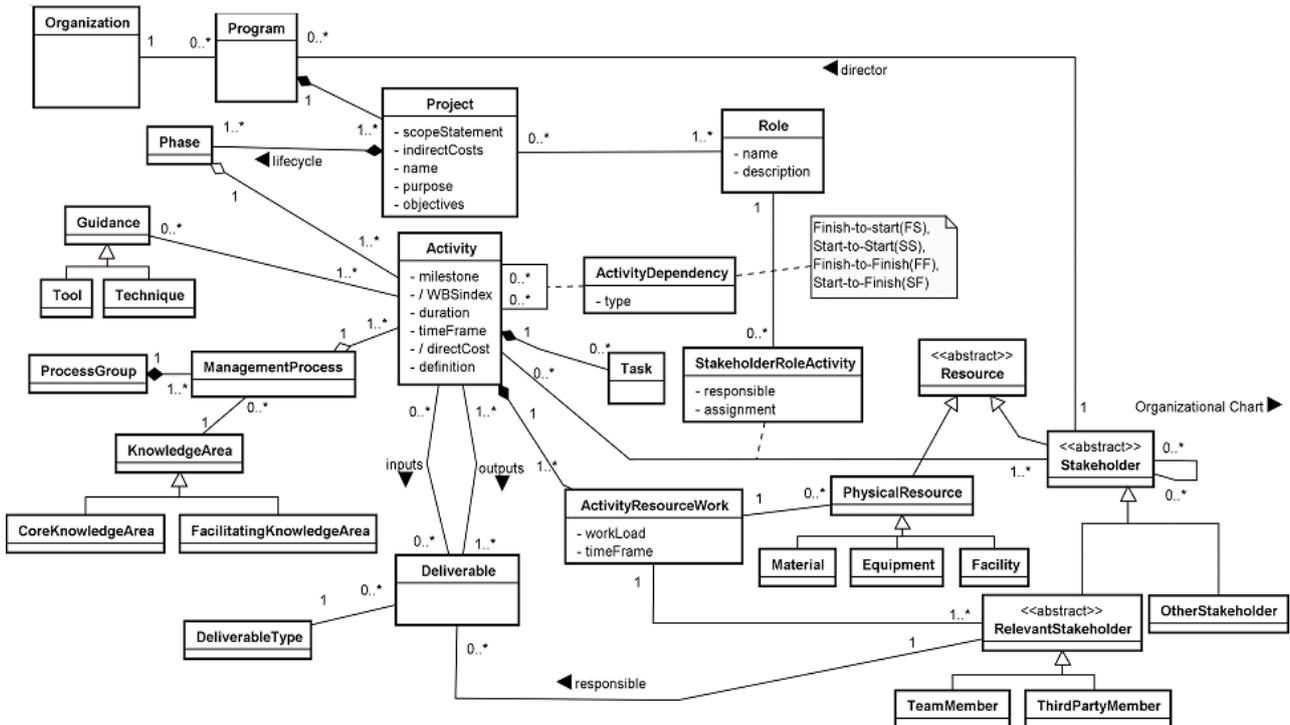



Figura 2. Metamodelo de gerência de projetos baseado no PMBOK Guide [CAL07]

Apesar do conceito de papéis (classe `Role`), os modelos de processos de desenvolvimento de software geralmente não abordam explicitamente as noções de recursos humanos e de outros tipos de entidades (tal como as partes interessadas) em conjunto com outros tipos de recursos (equipamentos, materiais e instalações de trabalho, por exemplo). Além disso, este modelo define uma classe chamada `RelevantStakeholder` que descreve os recursos que de fato atuam no projeto e que pode ser de dentro da empresa (classe `TeamMember`) ou de empresas terceirizadas (classe `ThirdPartyMember`). Neste modelo, `Stakeholder` é uma classe abstrata, e um ator não relevante é classificado como uma instância da classe `OtherStakeholder`.

Segundo este modelo, uma organização (classe `Organization`) contém uma coleção de programas (classe `Program`) que, por sua vez, são conjuntos de projetos (classe `Project`) dirigidos por uma dada parte interessada (classe `Stakeholder`). Um *stakeholder* pode assumir uma ou mais papéis (classe `Role`) em um projeto. Ao definir uma atividade, deve-se indicar as partes interessadas relacionadas a execução desta

atividade. Para cada associação, deve-se indicar qual o papel que cada parte interessada está cumprindo, bem como sua carga de trabalho (atributo `workload`). Pode-se também associar um recurso físico (classe `PhysicalResource`) usado para executar essa atividade.

O *PMBOK Guide* representa suas práticas em duas dimensões lógicas. Uma dimensão descreve as áreas de conhecimento (classe `KnowledgeArea`) enquanto que a outra dimensão descreve os processos gerenciais de um projeto (classe `ManagementProcess`), os quais estão contidos em cinco grupos de processo (classe `ProcessGroup`). As áreas de conhecimento são responsáveis por descrever as principais competências que os gerentes de projeto devem desenvolver e derivam as áreas centrais (classe `CoreKnowledgeArea`) e as de apoio (classe `FacilitatingKnowledgeArea`). Assim, cada atividade gerencial pertence a um processo gerencial, sendo também relacionada a uma área de conhecimento. Finalmente, as atividades podem ser subdivididas em tarefas (classe `Task`), e também pode depender de outras atividades pela classe `ActivityDependency`.

Os conceitos restantes reforçam que o ciclo de vida de um projeto é composto de fases, que por sua vez estão relacionados com as atividades. Atividades se relacionam com resultados como entradas e/ou saídas, e cada produto tem um tipo e uma parte interessada responsável única.

2.4.2 Rational Unified Process - RUP

2.4.2.1 Definição e Aspectos Relacionados

O *Rational Unified Process* (RUP) é um processo iterativo de desenvolvimento de software desenvolvido pela empresa *IBM Rational Software*, originado a partir do metamodelo SPEM [JAC01] [BEN09]. O RUP atua como um *framework* que pode ser adaptado e estendido de acordo com as características do processo de desenvolvimento de software da organização [RAT09]. Conforme a Figura 3, o RUP contém seus elementos em duas dimensões distintas: dinâmica e estática.

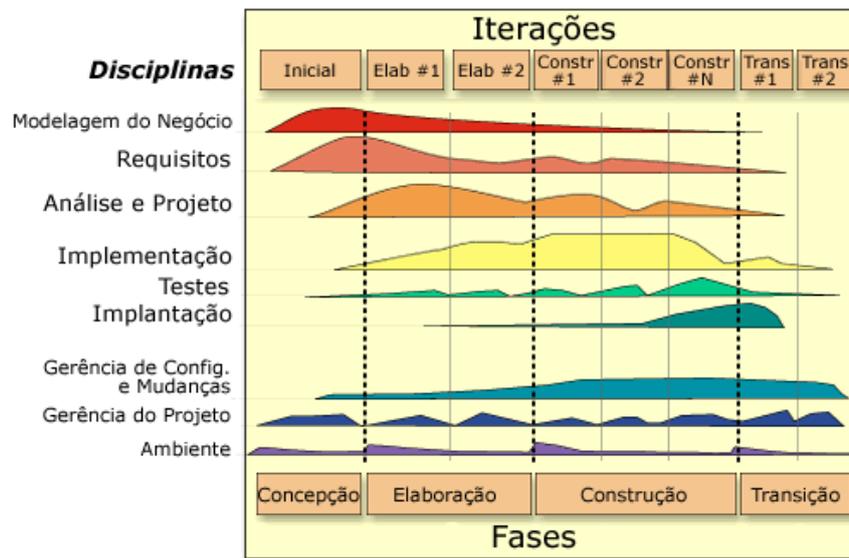



Figura 3. Visão geral da arquitetura do RUP [JAC01]

A dimensão dinâmica representa o tempo, mostrando os aspectos dinâmicos do processo através de ciclos, fases, iterações e milestones. De acordo com Jacobson, Booch e Rumbaugh [JAC01], o ciclo de vida do software é divido em quatro fases (concepção, elaboração, construção e transição) e, através de sua característica iterativa, cada fase é realizada com base no resultado da fase anterior, de maneira a refinar o sistema até o momento em que o produto final esteja completo. Cada fase possui um *milestone* e um conjunto de objetivos, que devem ser utilizados para servir como guia no momento de decidir quais atividades desempenhar e quais artefatos devem ser produzidos [BEN06]. Cada fase no processo RUP atravessará diversas iterações. Uma iteração é um laço completo do desenvolvimento tendo por resultado uma liberação (interna ou externa) de um produto executável, um subconjunto do produto final sob o desenvolvimento, que cresce incremental de iteração em iteração até se transformar no sistema final [JAC01].

A dimensão estática representa os aspectos estáticos descrevendo como os elementos do processo, atividades, disciplinas, artefatos e papéis, são agrupados em *workflows*. Assim, um processo deve ser capaz de descrever quem, como e quando uma determinada atividade está sendo desempenhada. Desta forma, o RUP é representado através de quatro elementos primários de modelagem: papéis (quem), atividades (como) artefatos (o quê) e fluxos (quando). Um papel define o comportamento e as responsabilidades de um indivíduo ou grupo de indivíduos que trabalham juntos como uma equipe [JAC01]. Os papéis são responsáveis pela execução das atividades do processo em cada uma das fases. Um indivíduo pode executar as tarefas de um ou mais

papéis, da mesma forma que um grupo de indivíduos pode executar as atividades de um mesmo papel. Uma atividade é uma unidade de trabalho que define como tarefas reais, atribuídas a um papel, devem ser executadas no contexto do projeto [RAT06]. Toda atividade deve ser atribuída a um papel. A atividade tem um propósito claro, normalmente expresso em termos de criar ou modificar artefatos.

Um artefato é um pedaço de informação que é produzido, modificado ou usado por um processo [RAT06]. Os artefatos são usados como entradas para executar uma atividade e são o resultado ou a saída de tais atividades. Os artefatos possuem diferentes formas de apresentação, tais como modelos, elementos de modelo, documentos, códigos fonte e executáveis. Deve-se ainda considerar que um artefato pode ser composto de outros artefatos. Os fluxos são as sequências de atividades, que através das iterações entre os papéis, produzem um resultado de valor observável [RAT06]. No RUP pode-se dividir os fluxos em dois grupos principais: fluxos centrais (*core workflows*), que são as disciplinas do processo, e os detalhes de fluxo (*workflow detail*), que são os fluxos internos de cada disciplina. Através da UML, um fluxo pode ser expresso como um diagrama de sequência, colaboração ou de atividade [BOO00]. É importante ressaltar que um fluxo não pode ser interpretado literalmente como sendo um conjunto de passos no qual um indivíduo irá executar de forma automática e mecânica. Cada disciplina é expressa através dos papéis (quem executa a tarefa), das atividades (como executar estas tarefas), e dos artefatos (o que a atividade consegue). Dessa forma, uma disciplina apresenta as atividades relacionadas de diferentes papéis em um fluxo de informação, assim, definindo como as atividades interagem com os artefatos.

2.4.2.2 Modelo de Referência para o RUP

De acordo com Jacobson, Booch e Rumbaugh [JAC01], o RUP utiliza os conceitos da linguagem UML para especificar e documentar os modelos de sistemas de software. O RUP apresenta um modelo semântico, ilustrado na Figura 4, contendo seus principais elementos e relacionamentos [PEP09]. Este modelo determina como os elementos do processo são organizados e quais as relações válidas entre estes elementos.

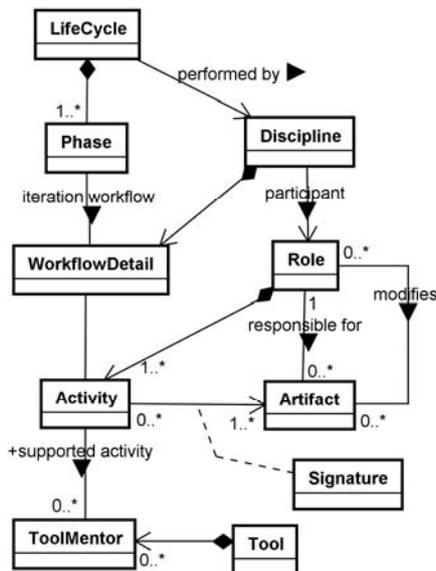



Figura 4. Modelo semântico do RUP [PEP09]

Na Figura 4, a classe `Lifecycle` representa o ciclo de vida de desenvolvimento de um software. Este conceito é particionado em um conjunto de quatro fases (classe `Phase`). A classe `Discipline` divide os elementos de processo em áreas de interesse. Um papel (classe `Role`) representa o elemento responsável por desempenhar atividades (`Activity`) para produzir ou modificar os artefatos (`Artifact`) do processo.

As informações de como os papéis devem colaborar entre si através de suas atividades são definidos pela classe `WorkflowDetail`. A classe `Artifact` descreve os tipos de produtos de trabalho que são produzidos ou consumidos no desempenho de atividades. Assim, a classe associativa `Signature` indica que um artefato é utilizado como entrada ou saída de uma atividade. A classe `Tool` descreve as ferramentas que podem ser utilizadas auxiliando a produção ou modificação de um artefato. Finalmente, a classe `ToolMentor` descreve o uso de ferramentas no contexto de algumas atividades.

2.4.3 Object-Oriented Process, Enviroment and Notation - OPEN

2.4.3.1 Definição e Aspectos Relacionados

O OPEN (*Object-Oriented Process, Enviroment and Notation*) é uma metodologia de desenvolvimento de software orientado a objetos mantido pelo *OPEN Consortium Group* [GRA97] [OPE09] [FIR02]. Ele pode ser definido como um *framework*, denominado *OPEN Process Framework* (OPF), que fornece um metamodelo extensível e que pode ser configurado para processos de desenvolvimento de software distintos [GRA97]. O OPEN

encapsula os conceitos e atividades relacionados ao negócio, qualidade, análise e reuso, e que são comuns a todo o processo de desenvolvimento de software, utilizando o paradigma de orientação a objetos.

Um processo é instanciado e customizado a partir do metamodelo do OPEN através da adição e remoção dos componentes de processo. Esta operação permite atender melhor as necessidades de uma organização em termos de tamanho, cultura, investimento e outras características, e envolve a escolha de atividades, tarefas, técnicas e configurações específicas para o negócio.

Uma unidade de processo (*process unit*) define um conjunto de atividades relacionadas que são executadas durante um projeto. Também define as entradas necessárias para gerar as saídas (*deliverables*), através da execução uma série de atividades [OPE02]. Além disso, o OPF define que as unidades do trabalho (*work units*) são componentes que modelam as operações que são executadas durante uma unidade de processo. O OPF considera os seguintes tipos de unidades de trabalho:

- Atividades: definem o que precisa ser feito (fluxo);
- Tarefas: definem o que fazer de forma coesa;
- Técnicas: definem como tarefas serão realizadas.

O ciclo de vida de desenvolvimento de projetos do OPEN descreve o tempo de duração em que o projeto é construído [FIR02]. O ciclo de vida é formado por um conjunto de atividades que produzem e consomem produtos de maneira gradativa durante a realização de tarefas. É um processo baseado em entregas onde cada estágio envolve uma ou mais entregas. Em todo estágio do ciclo de vida do OPEN muitas tarefas podem ser executadas e, para cada tarefa, diferentes técnicas podem ser utilizadas (Figura 5).

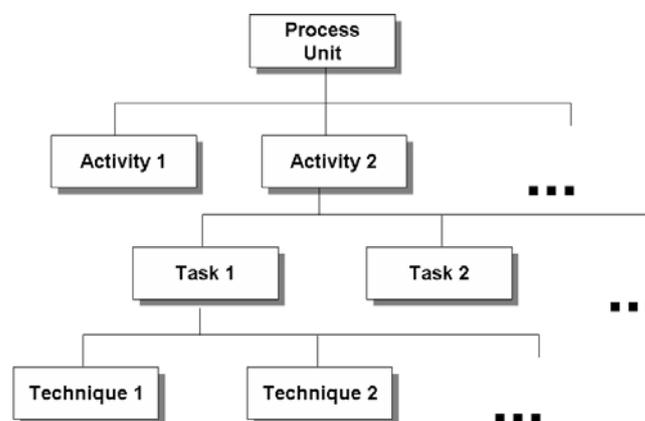



Figura 5. Relacionamento entre atividades, tarefas e técnicas [OPE09]

Cada atividade é definida como um conjunto de tarefas, que são a menor unidade de trabalho gerenciável. As tarefas são realizadas através do uso de técnicas. Dessa forma o OPEN inclui os conceitos de atividades com suporte ao ciclo de vida completo, além de tarefas e conjuntos de técnicas e artefatos.

2.4.3.2 Modelo de Referência para o OPEN

O *framework* do OPEN contém seu foco na interação cooperativa entre os produtores, suas unidades de trabalho e o que produzem [OPE09]. Dessa forma, o OPF reconhece as classes ilustradas na Figura 6 como sendo os componentes centrais de seu *framework*.

Neste modelo, a classe `Endeavor` refere-se ao componente que modela o esforço empreendido pelos produtores (`Producer`) que executam unidades de trabalho (`WorkUnit`) durante um ou mais estágios (`Stage`). Os componentes produzidos durante o desenvolvimento do projeto são definidos pela classe `WorkProduct`. A classe `Language` modela o tipo de linguagem utilizada para documentar e produzir os produtos do projeto.

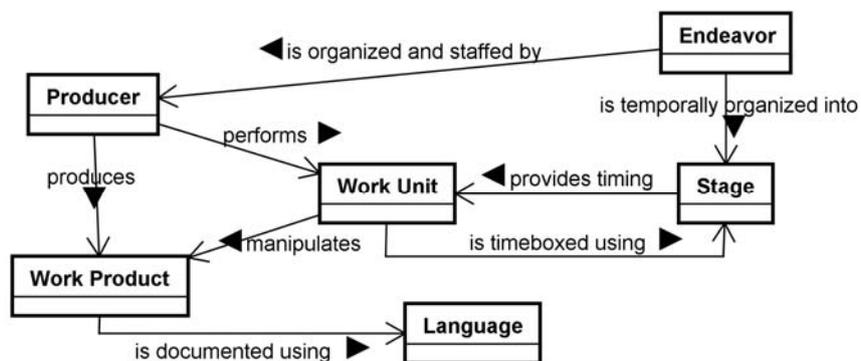



Figura 6. Componentes centrais ao framework do OPEN [OPF09]

Um produtor é o elemento responsável por produzir ou modificar – direta ou indiretamente – os artefatos do processo. Os produtores podem ser ferramentas ou pessoas definidas através de papéis. A classe `WorkUnit` consiste de um conjunto de operações coesas executadas pelo produtor no desenvolvimento seu trabalho, e podem ser classificadas como tarefas, técnicas, fluxos de trabalho e atividades. Finalmente, a classe `Stage` determina as divisões de intervalos de tempo do processo, sendo dividida em estágios com duração (fases) e instantâneos (*milestones*).

2.4.4 Software & Systems Process Engineering Meta-Model Specification - SPEM

2.4.4.1 Definição e Aspectos Relacionados

Nesta pesquisa o *Software Process Engineering Metamodel Specification* (SPEM) [OMG012] foi utilizado como principal processo de desenvolvimento de estudo. A escolha do SPEM deve-se ao fato deste ser considerado pelo*Object Management Group* (OMG) [OMG09] o principal *template* para modelos de processos de desenvolvimento de software. Além disto, sua utilização torna esta proposta aplicável a todos os processos de desenvolvimento de software que tiveram sua origem a partir deste metamodelo. Nesse sentido, esta seção fornece uma visão geral deste metamodelo. O interesse é descrever resumidamente como estão organizados seus elementos e relacionamentos.

O SPEM 2.0 separa a engenharia dos processos de desenvolvimento de software em dois momentos principais: criação de uma biblioteca de processos (*Method Library*), que armazenará o conteúdo do processo (*Method Content*) e a utilização deste conteúdo (*Process Structure*) em um processo de desenvolvimento de software. A Figura 7 fornece uma visão de como os conceitos definidos no SPEM 2.0 são posicionados para representar o conteúdo do processo (*Method Content*) e sua utilização (*Process Structure*).

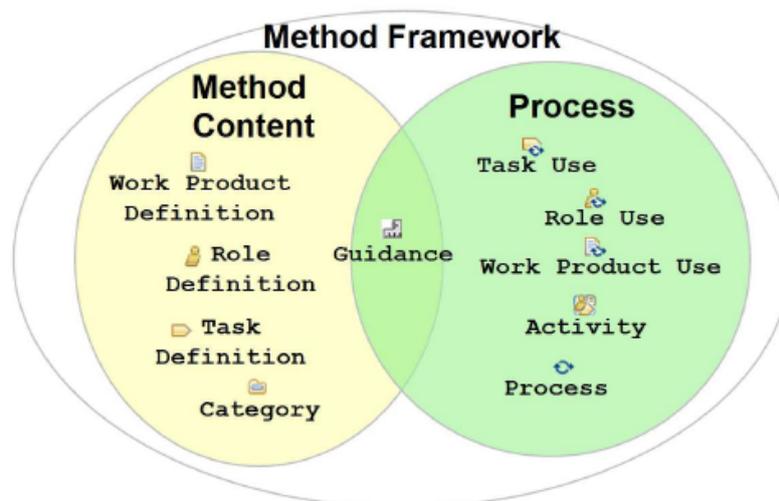



Figura 7 - Exemplo de divisão entre Method Content e Process Structure [OMG12]

*Method Content*é formado pelas definições dos produtos de trabalho, dos papéis e das tarefas. O*Process Structure* é a utilização do *Method Content* em um processo de desenvolvimento de software. Por fim, o elemento *Guidance*representa os guias,

checklists, exemplos ou *roadmaps* definidos para os processos de desenvolvimento de software.

2.4.4.2 Modelo de Referência para o SPEM

O SPEM é um metamodelo desenvolvido pelo OMG para a definição de processos de desenvolvimento de software e seus componentes. Esse metamodelo está estruturado em sete pacotes principais, conforme mostrado na Figura 8, os quais são compostos aplicando-se o mecanismo *package merge*[1] da UML. Tais pacotes são:

- *Core* - contém as metaclasses e abstrações para a construção da base do metamodelo;
- *Process Structure* - define os conceitos base para modelagem de processos representando sua estrutura estática.
- *Process Behaviour* - pacote que permite uma extensão do metamodelo do SPEM 2.0 para que a execução de um processo possa ser acompanhada.
- *Managed Content* - adiciona conceitos para documentação e descrição textual de um processo.
- *Method Content* - permite que os usuários do SPEM 2.0 criem uma biblioteca com conhecimento reutilizável e independente de processos para uso posterior.
- *Process with Methods* - define novos conceitos e altera outros conceitos já existentes nos pacotes anteriores para integrar processos definidos pelo pacote *Process Structure* com seus conteúdos, definidos pelo pacote *Method Content*.
- *Method Plugin* - define os conceitos necessários para criar, gerenciar e manter bibliotecas e processos de software.

O funcionamento e descrição das principais classes dos pacotes do SPEM 2.0, entretanto, são apresentados no Apêndice A.

---

[1]*Merge* é um relacionamento entre dois pacotes onde o conteúdo do pacote de destino é combinado com o conteúdo do pacote de origem através de especializações e redefinições, quando aplicáveis [OMG11].

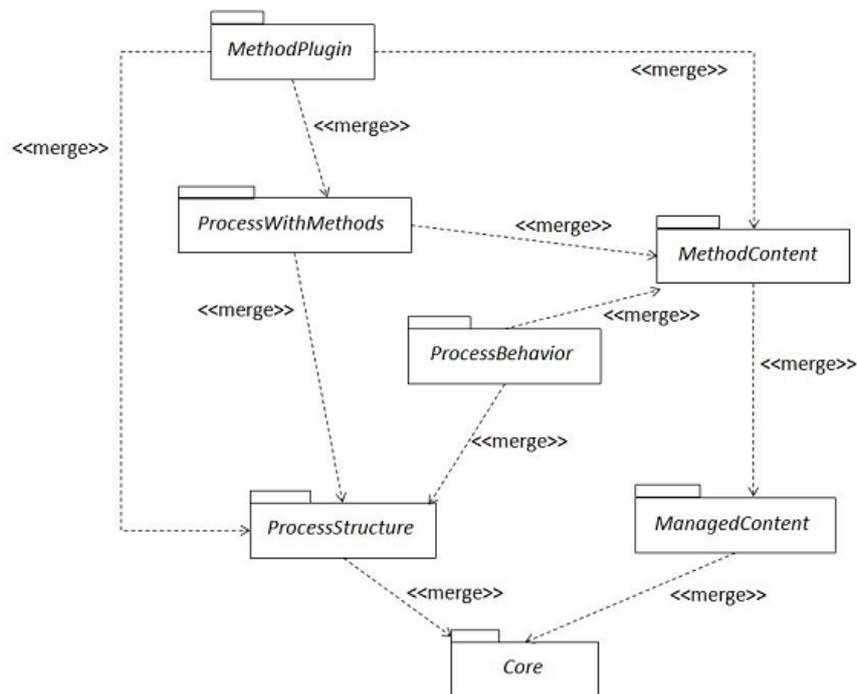



Figura 8. Camadas de metamodelagem propostas pela OMG

2.5 Considerações Finaisdo Capítulo

Este Capítulo apresentou os conceitos referentes à gestão de projetos e aos processos de software, objetivando demonstrar a importância de seu uso para as organizações de TI. Inicialmente, foi apresentado o ciclo de vida básico para um processo de software e, posteriormente, descritos os principais aspectos relacionados à gestão de projetos. Em seguida, foram apresentados os metamodelos estudados nesta pesquisa, a saber: PMBOK, SPEM, RUP e OPEN.

O próximo Capítulo apresenta os modelos de referência que foram usados como base para a elaboração do modelo de reconfiguração proposto nesta tese. Os modelos de referência foram comparados e combinados em um modelo integrado chamado SPIM.

3 INTEGRAÇÃO DOS PROJETOS DE SOFTWARE COM OS FLUXOS ORGANIZACIONAIS

3.1 Fluxos Organizacionais: Definição e Conceitos Relacionados

Projetos de software são muito dinâmicos e demandam recorrentes ajustes dos seus planos de projeto. Esses ajustes podem ser vistos como reconfigurações no cronograma das tarefas, na atribuição de recursos e de outros elementos do projeto. Neste contexto, um plano de projeto de software especifica e delimita o escopo do projeto, descreve possíveis riscos do projeto, define os recursos de hardware e software disponíveis, descreve a estrutura analítica do trabalho e a programação de projeto [LEE06].

Entretanto, deve-se considerar que o gerente de projetos pode não possuir todas as informações relevantes durante o planejamento e/ou a execução de um projeto de software e, consequentemente, poderão ocorrer interações com outros departamentos da organização (essa constatação já foi publicada pelo autor em [ROS08a]). O gerente de projetos, por exemplo, pode precisar contatar o setor de recursos humanos sobre a necessidade de contratação de pessoal para um determinado projeto de desenvolvimento de software. Percebe-se, dessa forma, que o fluxo de atividades de um projeto de software pode interagir com os demais fluxos de atividades comuns da organização (denominados aqui como sendo chamados de fluxos organizacionais).

Conforme ilustrado na Figura 9, ambos os tipos de fluxos de trabalho são executados em paralelo, possuem recursos próprios e podem influenciar nos prazos das atividades e custos do projeto de software. Devido a esta separação entre o fluxo de trabalho de um projeto de software e dos fluxos organizacionais, pode haver uma relação de dependência entre as atividades pertencentes a estes dois tipos de fluxos distintos. Por exemplo, a atividade de desenvolvimento de um site web (parte do fluxo de trabalho de projeto de software) pode depender a aquisição de um servidor web pelo departamento financeiro (parte do fluxo organizacional da empresa). A violação das

dependências entre as atividades de diferentes fluxos de trabalho pode resultar em distorções no planejamento de projetos de software.

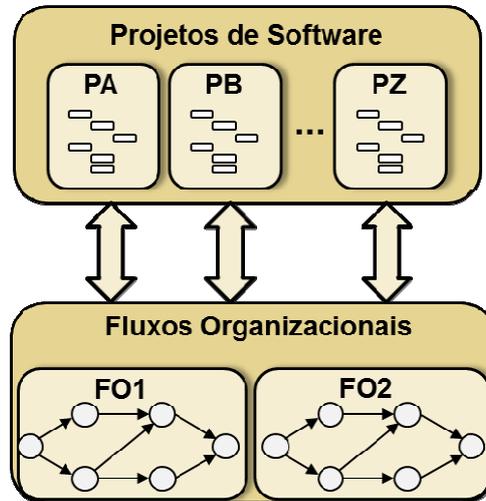



Figura 9. Relacionamento entre projetos de software e fluxos organizacionais

Observar-se, portanto, que alguns tipos de atividades gerenciais são inerentes ao processo e não aparecem no momento do planeamento inicial do projeto. São precisamente essas atividades (ou suas dependências) que mais frequentemente causam atrasos no cronograma e não são considerados na definição dos riscos do projeto. Muitas vezes o gerente de projetos somente percebe a necessidade de ter solicitado anteriormente uma informação de outro departamento da empresa no momento de executar uma determinada atividade do projeto que depende deste outro departamento (por exemplo: contratação de serviços terceirizados, a aquisição de equipamentos, a compra de passagens aéreas e reserva de hotéis para viagens de negócios, a coordenação logística para ministrar palestras ao público externo, a recepção de visitantes viajantes na empresa, entre outros exemplos). Consequentemente, existe a necessidade de uma solução que permita antecipar as necessidades advindas das áreas de apoio durante o planejamento do projeto de software. Esta solução deve considerar a complexidade de identificar a interdependência entre as atividades dos fluxos organizacionais e o fluxo de trabalho do projeto.

Portanto, para conciliar estas duas visões, o gerente de projeto deve lidar com questões tanto de gestão quanto de produção durante o planejamento e a execução de projetos, considerando não apenas as atividades que produzem resultados diretos no projeto de software, mas também as atividades que pertencem, de forma compartilhada a outros projetos, aos fluxos organizacionais da empresa.

Entretanto, os modelos de gerência de projetos e os processos de desenvolvimento de software atuais não apresentam uma solução que permita o planejamento de projetos de software considerando as interações do gerente de projetos com outros departamentos da organização. Esta solução deve considerar a complexidade de identificar a interdependência entre as atividades dos fluxos organizacionais e dos fluxos dos projetos.

3.2 A Necessidade de Integração

A gerência de projetos em um ambiente de desenvolvimento de software é definida como a gerência das pessoas e de outros recursos por um gerente de projetos a fim de planejar, analisar, projetar, construir, testar e manter um sistema de informação [SCH02]. Para cumprir estes objetivos, um gerente de projetos precisa de algum tipo de suporte, geralmente baseado em uma metodologia de gerência de projetos, que permita lidar com diferentes variáveis de projeto, responsabilidades e tarefas. Para este fim, existem diversas propostas na literatura ou práticas já realizadas nas empresas. Entretanto, a maioria dos modelos ou guias voltados para a gerência de projetos, tal como o *PMBOK Guide*, não se dirigem especificamente a processos de desenvolvimento de software.

Além disso, apesar do processo de desenvolvimento de software ser considerado um dos principais mecanismos responsáveis por gerenciar e controlar os projetos e produtos de software, muitos destes processos existentes apresentam carências no quesito de gerência de projetos. Tal fato é reforçado por Henderson-Sellers et al. [HEN00], onde é destacado que dois dos mais importantes processos de desenvolvimento de software, respectivamente o RUP, por sua absorção no mercado, e o OPEN, por sua contribuição no meio acadêmico, necessitam de maior suporte no quesito de gestão de projetos. Tanto o RUP quanto o OPEN auxiliam na execução das melhores práticas para o desenvolvimento de software. No entanto, o RUP, por exemplo, não cobre assuntos essenciais de gestão de projetos, tais como a gerência de pessoas e a gerência de contratos. Em contraste, o OPEN apresenta um conjunto de atividades e técnicas que contemplam diferentes áreas da gerência de projetos, tais como qualidade, estimativas de custos e métricas de gerenciamento. Todavia, ambos os modelos mostraram-se deficientes em áreas essenciais de conhecimento da gerência de projetos, particularmente, a gerência de aquisição, de comunicação e de pessoas.

A fim de se obter um processo mais detalhado para o gerenciamento de projetos software, é necessário aplicar os conhecimentos de gestão de projetos aos processos de desenvolvimento do software. Portanto, se por um lado o PMBOK pode fornecer uma perspectiva gerencial da solução, a visão sobre a produção deve ser obtida a partir de um modelo de processo de desenvolvimento de software. Segundo [PER07], processos de software definidos a partir de um metamodelo geralmente oferecem um alto grau de formalismo e melhor suporte para consistência e customização, uma vez que os conceitos que formam sua base são explicitamente definidos.

Pesquisas anteriores apresentaram resultados interessantes, mas uma íntima integração da gerência de projetos e dos processos de software com resultados práticos ainda é uma questão em aberto [HEN00], [HEN01], [REH07], [SCH02]. Consequentemente, faz-se necessário mais estudo para uma solução que permita um melhor nível de integração entre os conceitos e modelos para estas duas áreas de conhecimento. Dessa forma, em seguida são listadas asnecessidades identificadas a partir da literatura examinada nesta pesquisa.

1) Acesso às informações pertencentes aos outros departamentos da organização

Ao realizar planejamento de um projeto de software, o gerente de projetos pode necessitar interagir com outros departamentos da organização a fim de obter informações relevantes para o projeto (contatar o setor de recursos humanos sobre a necessidade de contratação de pessoal, por exemplo). Além disso, estes outros setores da organização são responsáveis por atualizar as informações sobre custos e prazos destas atividades de apoio ao projeto de software. Com o objetivo de obter estas informações, percebe-se que o fluxo de atividades de um projeto de software poderá interagir com os fluxos organizacionais. Consequentemente, o gerente de projetos precisa de uma solução que permita o acesso às informações dos fluxos organizacionais durante a elaboração do planejamento de projetos de software.

2) Identificação das relações de dependência entre as atividades dos fluxos organizacionais e dos projetos de software

As atividades pertencentes a um fluxo organizacional não são exclusivas de um projeto de software específico, mas de um fluxo comum da empresa que é compartilhado pelos projetos em andamento. Neste instante, percebe-se que há uma dissociação entre o fluxo de atividades de um projeto de software e os demais fluxos de atividades de suporte

ao projeto da organização. Durante o planejamento de atividades do projeto, por exemplo, o gerente de projetos informa o setor de recursos humanos sobre a necessidade de contratação de um analista de testes. Neste caso, constata-se a existência de uma relação de dependência entre as atividades do projeto (tais como, a modelagem dos casos de teste) com as atividades pertencentes ao fluxo de trabalho do setor de recursos humanos referentes à contratação do profissional requerido para executar a atividade do projeto de software.

A dificuldade para identificar a interdependência dos fluxos de trabalho da empresa e do projeto de software durante o planejamento do projeto pode resultar, por exemplo, no aumento dos custos e em atrasos nos prazos do projeto. Assim, percebe-se a necessidade de identificação das relações de dependência entre as atividades pertencentes a estes dois tipos de fluxos de trabalho, permitindo a antecipação das necessidades advindas das áreas de apoio da organização durante o planejamento de projetos de software.

3) Capacidade de minimizar distorções no planejamento de projetos (tais como, o aumento dos custos e atrasos nos prazos do projeto)

Conforme mencionado anteriormente, os fluxos organizacionais e do projeto de software são executados de forma distinta. Além disso, as atividades pertencentes aos fluxos organizacionais utilizam recursos não alocados diretamente ao projeto de software. Estes recursos podem influenciar tanto nos prazos das atividades quanto nos custos do projeto de software. Ao realizar o planejamento de um projeto de software, por exemplo, o gerente de projetos identifica a necessidade de contratação de um analista de banco de dados. Esta empresa possui uma política de admissão de profissionais que exige a realização de exames médicos antes da contratação. Dessa forma, o gerente de projetos faz uma previsão de quando poderá utilizar este novo recurso no projeto. Entretanto, caso o médico responsável pelo exame de admissão precise ficar ausente por alguns dias, este atraso poderá impactar negativamente no cronograma do projeto de software em questão.

Dessa forma, o gerente de projetos precisa de um suporte que o auxilie a evitar distorções no planejamento de projetos (tais como, o aumento dos custos e atrasos nos prazos do projeto) pela desconsideração de que as atividades de apoio pertencentes aos fluxos organizacionais utilizam recursos que não são alocados diretamente ao projeto de software.

4) Auxílio na identificação e mensuração dos custos indiretos do projeto

O gerente de projetos precisa lidar tanto com assuntos gerenciais quanto técnicos durante o planejamento e a execução de atividades. Ele deve considerar, durante o planejamento de projetos de software, tanto as atividades que produzem um resultado significativo no contexto de um projeto de software quanto as atividades que pertencem exclusivamente aos demais fluxos de atividades de suporte ao projeto da organização. Dessa forma, a distinção explícita entre as atividades pertencentes aos fluxos organizacionais e as de um projeto de software específico permite a identificação e a mensuração dos custos indiretos do projeto de software advindos das atividades de apoio da organização.

Observa-se, dessa forma, que o gerente de projetos precisa de algum tipo de suporte para lidar tanto com assuntos gerenciais quanto técnicos durante o planejamento e a execução de atividades, levando em consideração não só as atividades do projeto de software, mas também as atividades que pertencem de forma compartilhada aos demais fluxos de atividades de suporte aos projetos da empresa.

A seção a seguir apresenta os modelos intermediários de integração entre os aspectos advindos da gerência de projetos com aqueles advindos dos modelos de processos de desenvolvimento de software.

3.3 Modelos Intermediários de Integração

A *Meta Object Facility* (MOF) [OMG11] é uma estrutura padrão de gerenciamento de metadados desenvolvida para permitir a interoperabilidade de modelos e sistemas de metadados [OMG11]. O conceito central na arquitetura MOF é a noção de um modelo. Assim, um modelo é projetado para um determinado domínio de aplicação e descreve suas entidades e suas relações entre si. Cada modelo tem uma semântica de metadados, que é definida através do significado dos seus elementos em um determinado contexto, e uma sintaxe (concreta), que pode ser simbólica, textual ou gráfica. A sintaxe abstrata de um modelo descreve a sua estrutura. Este modelo de arquitetura proposto pela OMG é composto de quatro camadas ou níveis (conforme ilustrado na Figura 10).

Quanto aos sistemas de metadados, podem-se identificar três tipos de modelos [OMG11]:

- Esquemas de Metadados: definem os elementos de metadados (por exemplo, criador, assunto, trabalho). Resumidamente, estes elementos podem ser

considerados como componentes de um modelo de metadados, que descreve um determinado domínio de aplicação. Pode-se citar como exemplos deste tipo de modelo o *Dublin Core* (DC) e a *Learning Object Metadata* (LOM);

- Linguagens para a definição de esquemas: definem os modelos de metadados mencionados anteriormente. Pode-se citar como exemplos deste tipo de modelo o *XML Schema* e OWL. Estes fornecem um conjunto de primitivas de linguagem (por exemplo, *complexType*, *Class*, etc.) com uma definição semântica precisa e uma notação sintática correspondente. Uma vez que existe uma correspondência rigorosa entre os elementos de um modelo de metadados e as primitivas da linguagem, uma linguagem de definição de esquemas pode ser concebida como um metamodelo de um modelo de metadados. Este tipo de modelo pode ser considerado como um metamodelo de metadados.
- Linguagens de modelagem universais: onde as próprias linguagens de definição de esquemas são definidas. Este tipo de modelo pode também ser considerado como sendo um meta-metamodelo de metadados.

A especificação MOF oferece uma definição para estes três tipos de modelos, e organiza- se em três níveis: M1 é o nível que contém os elementos do modelo para uma aplicação particular. A definição de um processo de desenvolvimento (por exemplo, RUP e OPEN) aparece no nível M1. As linguagens de definição de esquemas residem na camada M2. O metamodelo SPEM 2.0, por exemplo, aparece no nível M2 e serve de *template* para o nível M1. No nível mais alto, a M3, pode-se encontrar as linguagens universais de modelagem (por exemplo, definição de construtores e tipos primitivos) em que os sistemas de modelagem são especificados. O nível M0 do MOF descreve os objetos reais de um domínio (por exemplo, a criação de um processo de software para uma determinada empresa, baseado no RUP). A Figura 10 ilustra as camadas MOF e as correspondências entre os modelos de cada camada.

Segundo esta estrutura, um modelo definido em uma camada superior define a linguagem a ser usada na camada inferior seguinte [OMG11]. Nesta pesquisa, realizou-se a integração de modelos que pertencem às camadas M1 e M2 do MOF.

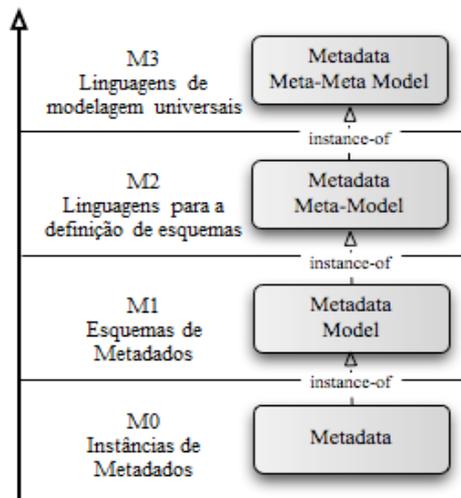



Figura 10. Hierarquia de metaníveis do MOF

O estudo detalhado dos modelos do PMBOK, SPEM, RUP e OPEN permitiu identificar como são organizados e quais as relações válidas entre os elementos de cada metamodelo. Através da integração entre a gerência de projetos e os processos de desenvolvimento de software foi possível identificar as principais características e discrepâncias entre os elementos de tais metamodelos. Conforme citado anteriormente, o metamodelo de referência do PMBOK [CAL07] inclui os elementos necessários para a gerência de projetos de software, enquanto que os conceitos de processos de desenvolvimento de software são obtidos pelo SPEM, RUP e OPEN.

O critério de integração entre estes modelos seguiu um conjunto de regras identificado por Callegari e Bastos [CAL07], que afirma que ao se realizar uma integração entre dois modelos, as seguintes situações podem ocorrer:

- Uma sobreposição de conceitos (duas classes com o mesmo conceito em cada modelo): neste caso, pode-se transformar e unir estas duas classes em um único conceito dentro de um pacote comum;
- Relação entre conceitos (uma classe de um dos modelos se relaciona com alguma outra classe de outro modelo, mas estas classes não representam exatamente o mesmo conceito): devem-se manter as classes em seus modelos originais e criar uma associação entre elas; e
- Conceitos independentes (classes com conceitos independentes e distintos): deve-se deixar cada classe em seu próprio pacote.

A proposta de integração entre a gerência de projetos e os processos de desenvolvimento de software apresentada neste trabalho é constituída de três pacotes: um para os conceitos de gerência de projetos (“PMBOK”), outro para os relacionados aos

processos de desenvolvimento de software (“SPEM”, “RUP” e “OPEN”) e, finalmente, um pacote comum (“Common”) que une os conceitos que ocorrem em ambos os modelos.

3.3.1 Modelo Integrado para o PMBOK e o RUP

O estudo detalhado dos modelos do PMBOK e RUP permitiu, dessa forma, identificar como são organizados e quais as relações válidas entre os elementos de cada modelo. Através da integração entre o modelo do PMBOK com o modelo do RUP foi possível identificar as principais características e discrepâncias entre os elementos de tais modelos. O conjunto adicional de classes e relacionamentos propostos ao modelo integrado para o PMBOK e o RUP baseou-se nos estudos realizados em [PMI08], [SCH02], [BEN09], [JAC01], [RAT09] e [PEP09].

Para efeito da discussão proposta nesta pesquisa, observa-se que os conceitos do PMBOK são, em sua maioria, representados por textos descritivos. Entretanto, objetivando comparar dois modelos e, posteriormente, realizar a integração entre eles, deve-se representá-los sob estruturas compatíveis – neste documento será utilizado o metamodelo previamente desenvolvido porCallegari e Bastos [CAL07], usando notação UML.

É importante ressaltar que alguns conceitos relacionados à gerência de projetos que estão contidos nos processos de desenvolvimento de software estudados foram propositadamente deslocados para o pacote de classes gerenciais (pacote PMBOK) com o objetivo de deixar mais explícita a classificação dos conceitos de gerência de projetos e do processo de desenvolvimento de software. Também optou-se por manter os conceitos na língua inglesa para facilitar a comparação com os modelos originais.

Neste modelo, vide Figura 11, a classe `Organization` representa uma empresa que se organiza por programas (classe `Program`). Os programas são grupos de projetos (classe `Project`) designados a alcançar um objetivo estratégico. As organizações geralmente dividem os projetos em várias fases (classe `Phase`) visando um melhor controle gerencial.

Os recursos necessários para um projeto são explicitamente descritos no subpacote `Resources`. Sendo assim, pessoas, equipamentos e locais são representados pela classe `Resource`. Estes recursos são divididos em recursos ativos (classe `Stakeholder`) e não ativos (classe `PhysicalResource`). Os

*stakeholders*correspondem às pessoas e organizações cujos interesses são afetados pelo projeto [PMI08]. De acordo com Schwalbe [SCH02], um projeto deve ter apenas um recurso responsável por direcionar e fundamentar o projeto. A importância da correta identificação dos principais *stakeholders* deve-se ao fato de que o sucesso do projeto depende, entre outros fatores, da capacidade de atender as necessidades e expectativas dos *stakeholders*. Dessa forma, a classe associativa `ProjectStakeholder` informa, explicitamente, qual é o *stakeholder* responsável por um determinado projeto. Também, define o nível de interesse e o nível de influência deste *stakeholder* no projeto.

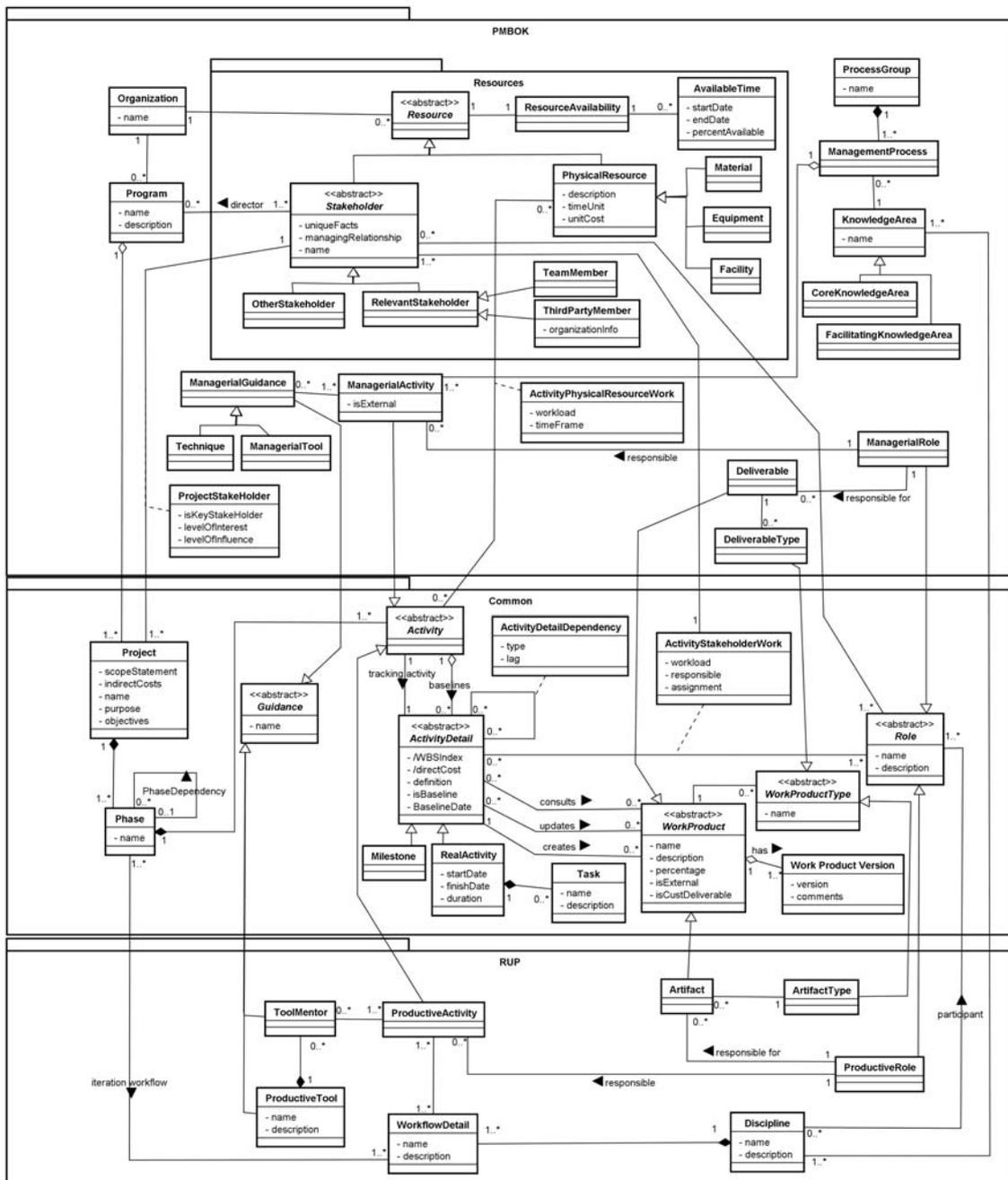



Figura 11. Metamodelo de integração PMBOK+RUP [ROS08a]

Conforme salientado anteriormente, os conceitos de gerência de projetos que existem no RUP foram atribuídos ao pacote gerencial (neste caso, o PMBOK) a fim permitir uma distinção explícita entre os conceitos produtivos e gerenciais. Por exemplo, no RUP um papel interage com ambas as tarefas gerenciais e produtivas, mas no metamodelo integrado foram definidas, anteriormente, as classes especializadas `ProductiveRole` (pacote `RUP`) e `ManagerialRole` (pacote `PMBOK`). Assim, a classe `Role` pertence ao pacote comum. Considerando que uma atividade pode ser derivada em atividade gerencial (classe `ManagerialActivity`) ou produtiva (classe `ProductiveActivity`), foi necessário duplicar o relacionamento que existia entre as classes `Role` e `Acticity` (denominado `responsible`) de maneira a respeitar a divisão entre gerência de projetos e desenvolvimento de software que está sendo proposta nesta pesquisa. Além disso, por este mesmo motivo foi necessário duplicar o relacionamento que existia entre as classes `Role` e `Workproduct` (denominado `responsiblefor`). Dessa forma, foi explicitamente adicionado ao metamodelo o conceito que apenas um papel gerencial é responsável por um produto de trabalho gerencial (classe `Deliverable`) e que apenas um papel produtivo é responsável por um produto de trabalho produtivo (`Artifact`).

O subpacote `Resources` também contém informações sobre a disponibilidade de cada recurso ao atribuí-lo às atividades, mesmo que realizado de forma manual ou automática, através da classe `ResourceAvailability`. Esta classe permite automatizar os processos de alocação de recursos em projetos de software. Foi adicionado ao modelo a classe `AvailableTime`, que informa quando um recurso está disponível. Esta classe contém atributos que informam a data inicial, data final e o percentual de alocação de um recurso a uma determinada atividade. É importante ressaltar que esta disponibilidade é independente do projeto, mas não é independente da organização em que atua. Paralelamente, a definição da carga de trabalho (atributo `workload`) dos recursos físicos (ao associar-se a diferentes atividades) é representada pela classe `ActivityPhysicalResourceWork`.

Este metamodelo define que a classe `Activity` pode ser especializada como atividade produtiva (`ProductiveActivity`) ou atividade gerencial (`ManagerialActivity`), relacionadas ao pacote relacionado aos processos de desenvolvimento de software (RUP, neste caso) e ao pacote `PMBOK` respectivamente.

Uma atividade produtiva representa uma unidade de trabalho, desempenhada por um papel produtivo, que produz um resultado significativo no contexto de um projeto de software (por exemplo, a modelagem do banco de dados). As atividades gerenciais, entretanto, podem pertencer tanto ao fluxo de desenvolvimento de software quanto aos fluxos organizacionais. Esta distinção é possível através do atributo denominado `isExternal` da classe `ManagerialActivity`, de maneira que `isExternal=false` define uma atividade gerencial como sendo pertencente ao projeto de software enquanto que `isExternal=true` refere-se a uma atividade gerencial de apoio da organização. Consequentemente, neste modelo identificam-se três tipos de atividades: **produtivas**, **gerenciais** e **gerenciais de apoio**. Seguindo esta nomenclatura, a atividade de organizar e conduzir uma reunião de acompanhamento do projeto é um exemplo de uma atividade gerencial que pertence exclusivamente ao projeto de desenvolvimento de software. Em contrapartida, a atividade de contratar um administrador de banco de dados é um exemplo de uma atividade gerencial que pertence exclusivamente aos fluxos organizacionais (neste caso, esta atividade é realizada pelo setor de recursos humanos).

Adicionalmente, cada atividade pode pertencer a um ou mais *baselines*. Em cada geração da *baseline*, uma atividade deve manter os relacionamentos com os papéis e produtos de trabalho. Assim, objetivando-se manter estes relacionamentos das atividades com outras entidades do modelo durante a geração de baselines, decidiu-se assumir que a classe `Activity` conterá uma agregação de uma ou mais instâncias da classe `ActivityDetail`. Dessa forma, foi adicionado ao modelo a classe `ActivityDetail` objetivando-se manter estes relacionamentos das atividades com outras entidades do modelo durante a geração de *baselines*. Assim, a classe `ActivityDetail` (pacote `Common`) foi definida como responsável por manter estes relacionamentos, enquanto que a classe `Activity` foi definida como sendo uma agregação de uma ou mais classes `ActivityDetail`. Dessa forma, enquanto que o relacionamento denominado `Baselines` permite que uma atividade pertença a uma ou mais *baselines*, o relacionamento `TrackingActivity` diferencia a atividade atual daquelas pertencentes às *baselines*. Também foi adicionado ao pacote `Common` a classe `RealActivity` (representa uma atividade que tem tempo de execução) e a classe `Milestone` (é uma atividade que não tem tempo de duração). Também, uma atividade pode possuir um

tempo de duração definido e ser atribuída a um papel (classe RealActivity) ou pode não ter um tempo de duração (classe Milestone).

Os *stakeholders* podem desempenhar diversos papéis (classe Role) durante a execução das atividades do projeto. Assim, para cada associação entre um papel e uma atividade (representado pela classe associativa ActivityStakeholderWork) deve haver também uma associação dessa atividade com um *stakeholder* capaz de desempenhar aquele papel. Além disso, como o conceito de papéis (classe Role) aparece em ambos os modelos, estes foram divididos em papéis gerenciais (classe ManagerialRole) e papéis produtivos (classe ProductiveRole), tal como ocorreu com as atividades. Cabe ressaltar que os conceitos de gerência de projetos que existem no RUP foram atribuídos ao pacote gerencial (neste caso, o PMBOK) a fim permitir uma distinção explícita entre os conceitos produtivos e gerenciais. Por exemplo, no RUP um papel interage com ambas as tarefas gerenciais e produtivas, mas no metamodelo integrado foram definidas, anteriormente, as classes especializadas ProductiveRole (pacote RUP) e ManagerialRole (pacote PMBOK). Assim, a classe Role pertence ao pacote comum. Considerando que uma atividade pode ser derivada em atividade gerencial (classe ManagerialActivity) ou produtiva (classe ProductiveActivity), foi necessário duplicar o relacionamento que existia entre as classes Role e Acticity (denominado responsible) de maneira a respeitar a divisão entre gerência de projetos e desenvolvimento de software que está sendo proposta nesta pesquisa.

Em relação à gerência de projetos, o *PMBOK Guide* representa suas práticas em duas dimensões lógicas. Uma dimensão descreve as áreas de conhecimento (classe KnowledgeArea) enquanto que a outra dimensão descreve os processos gerenciais de um projeto (classe ManagementProcess), os quais estão contidos em cinco grupos de processo (classe ProcessGroup). As áreas de conhecimento são responsáveis por descrever as principais competências que os gerentes de projetos devem desenvolver e derivam as áreas centrais (classe CoreKnowledgeArea) e as de apoio (classe FacilitatingKnowledgeArea). Assim, cada atividade gerencial pertence a um processo gerencial, sendo também relacionada a uma área de conhecimento.

Neste modelo, o pacote RUP define uma disciplina (classe Discipline) como sendo a divisão de elementos de processo em áreas de interesse. Cada disciplina é composta por um ou mais fluxos de trabalho (classe WorkflowDetail). Os fluxos de

trabalho definem como os papéis produtivos devem colaborar entre si através de suas atividades.Um produto de trabalho (classe `WorkProduct`) pode ser classificado como um produto gerencial (classe `Deliverable`) ou produtivo (classe `Artifact`), o qual deve estar associado a um tipo de produto (classes `DeliverableType` e `ArtifactType`, respectivamente). Cabe salientar que o modelo faz distinção das possíveis relações entre uma atividade e um artefato (criar/atualizar/consultar). Isto é importante para apoiar a consistência do modelo a partir de regras, tais como as apresentadas naTabela 8, que são detalhadas mais adiante no Capítulo 6.

3.3.2 Modelo Integrado para o PMBOK e o OPEN

O desenvolvimento do metamodelo PMBOK+RUP ajudou a identificar como as classes estão organizadas e as relações válidas entre os elementos de cada modelo. Dessa forma, considerando a absorção acadêmica do modelo OPEN, realizou-se um estudo de viabilidade para o desenvolvimento de um modelo integrado com este metamodelo. Assim, o metamodelo de integração para o PMBOK e o OPEN, denominado PMBOK+OPEN, apresenta uma estrutura similar ao apresentado na seção anterior, substituindo apenas o pacote referente ao processo de desenvolvimento de software. Assim, as classes dos pacotes `PMBOK` e `Common` são as mesmas que as apresentadas no metamodelo PMBOK+RUP. O resultado desta integração foi publicado em [ROS12b].

Os dois processos de desenvolvimento de software, porém, possuem características particulares que são refletidas em classes distintas e em diferentes relacionamentos com os pacotes `PMBOK` e `Common`. Desta forma, faz-se necessário uma análise comparativa entre as classes dos metamodelos do RUP com o OPEN.

3.3.2.1 Análise Comparativa entre os Metamodelos RUP e OPEN

A análise dos metamodelos destes processos de desenvolvimento de software (vide Tabela 1) permite identificar os pontos de conformidade entre os elementos centrais do RUP e do OPEN. Esta análise comparativa foi baseada em estudos realizados em [FIR02] e [OMG12] e contribuem para o estudo apresentado em [ROS06].

No RUP, a classe `Tool` descreve as ferramentas que ajudam a produção ou modificação de um artefato. O metamodelo OPEN também contém uma classe chamada `Tool`, que é uma subclasse da classe `Producer`, e representa um software usado para

criar ou modificar as versões de produtos de trabalho. Neste caso, observou-se que a classe `Tool` no RUP é equivalente à classe `Tool` no OPEN.

Tabela 1: Análise Comparativa entre os principais conceitos do RUP e do OPEN

RUP	OPEN
Tool	*Producer (Tool)*
Tool Mentor	*Work Unit (Technique)*
Artifact	*Work Product*
Activity	*Work Unit (Task)*
Role	*Producer (Role)*
Discipline	*Activity*
Lifecycle	*Stage (Lifecycle)*
Phase	*Stage(Phase)*
Workflow Detail	*Activity*
Signature	-
-	*Endeavor*
-	*Language*

Existe uma similaridade de conceitos provenientes da classe `ToolMentor` no RUP e a classe `Technique` no OPEN. A classe `ToolMentor` é responsável pela orientação sobre como as atividades são realizadas utilizando um instrumento particular. A classe `Technique`, uma subclasse da classe `Work Unit`, é responsável por determinar como realizar uma ou mais atividades, fluxos de trabalho e tarefas por um produtor (classe `Producer`).

De acordo com o RUP, a classe `Artifact` descreve os tipos de produtos de trabalho que são produzidos e modificados durante o projeto. A classe de `Work Product` do modelo OPEN representa tudo o que é produzido, utilizado, modificado ou destruído durante a realização de uma ou mais unidades de trabalho por um ou mais produtores. Neste caso, existe uma pequena diferença entre o RUP e o OPEN sobre a relação destas duas classes com um papel (classe `Role`). No RUP, um artefato deve ser de responsabilidade de apenas um papel e pode ser modificado por qualquer um ou vários papéis. No OPEN, um produto de trabalho deve estar relacionado com um ou mais produtores. Assim, a classe `Artifact` do RUP é equivalente à classe de `Work Product` do OPEN.

A definição da classe `Activity` no RUP satisfaz a definição da classe `Task` no OPEN. A classe `Activity` representa uma unidade de trabalho que produz um resultado significativo para o projeto, enquanto a classe `Task` representa um trabalho específico que produz ou modifica um ou mais produtos de trabalho. Ambos os modelos RUP e

OPEN usam o termo Role para definir quem é responsável pela execução das atividades e por produzir ou modificar produtos de trabalho. No OPEN, a classe Role é uma subclasse de Producer.

No RUP, a classe Discipline é responsável pela divisão dos elementos do processo em áreas de interesse. Cada disciplina é composta de um ou mais fluxos de trabalho. A classe Workflow Detail agrupa atividades relacionadas e define como os papéis devem trabalhar em conjunto para alcançar objetivos específicos do processo. Esta classe determina a sequência e interdependência entre as atividades pertencentes a uma disciplina. No OPEN, no entanto , o conjunto de elementos relacionadoscom uma única atividade (tais como os produtores, os produtos de trabalho e unidades de trabalho), que são parte de um único campo de conhecimento, é definido pela classe Activity. Assim, no OPEN uma disciplina está organizada em torno de uma única atividade, que pode conter várias tarefas. Além disso, esta classe é composta de um conjunto de tarefas, agrupadas de acordo com um objetivo comum e produz um conjunto de produtos interligados, onde as relações de dependência entre as atividades podem ser definidas através de pré-condições e pós-condições. Consequentemente, a classe Activityno OPEN engloba os conceitos de classe Workflow Detaile Discipline do RUP.

De acordo com o RUP, a classe Lifecycle define o ciclo de vida de desenvolvimento do software. O OPEN propõe uma divisão entre os ciclos de vida de produtos e processos através das classes Business Engineering Cycle, Life Cycle e Development Cycle. Assim, a classe Life Cycle do OPEN é compatível com a classe de Lifecycle do RUP, pois representa o conjunto de fases em que um único sistema, aplicativo ou componente principal que é produzido ou utilizado.

A classe de Signature no RUP contém dois atributos mutuamente exclusivos que indicam se um atributo é usado para entrada ou saída de uma determinada atividade. Neste caso, não foi identificada uma classe semelhante no metamodelo OPEN.

A classe Endeavor no modeloOPEN representa o esforço empreendido pelos produtores durante a execução das unidades de trabalho. Este conceito não foi encontrado no metamodelo RUP.Finalmente, a classe Language (que se refere ao tipo de linguagem utilizada para documentar e produzir o projeto), não demonstra conformidade com qualquer classe do RUP.

3.3.2.2 Intergrando os metamodelos PMBOK e OPEN

A integração com o OPEN permitiu tanto a confirmação quanto a adequação dos conceitos propostos no modelo final. O conjunto de classes do pacote OPEN é apresentado a seguir. Cabe salientar que a Figura 12 apresenta apenas os componentes centrais (classes `Producer`, `WorkUnit`, `WorkProduct`, `Stage e Language`) do *framework* do OPEN. Estes componentes representam classes abstratas e derivam num conjunto maior de subclasses (algumas destas subclasses são ilustradas no pacote OPEN).

A análise das classes e relacionamentos do metamodelo de integração PMBOK+OPEN baseou-se nos estudos realizados em [PMI08], [SCH02], [OPE09], [FIR02] e [GRA97] e tiveram como objetivo evitar possíveis inconsistências no metamodelo.

Neste modelo (vide a Figura 12), a classe `Endeavor` representa o esforço empreendido pelos produtores durante o desenvolvimento do projeto. Esta classe possui como responsabilidade desenvolver e/ou manter um ou mais produtos e serviços relacionados ao esforço empreendido. O OPEN define que a classe `Enterprise` representa o nível mais elevado de esforço, consistindo em uma coleção de programas relacionados que são controlados como uma única unidade. Possui as seguintes subclasses: `Enterprise`, `Program` *e* `Project`. A classe `Enterprise` representa o esforço de mais alto nível, composto por um conjunto de programas relacionados que são gerenciados como uma única unidade.

A classe `Producer` descreve um componente central do OPF que fornece serviços e produz, direta ou indiretamente, as versões de produtos de trabalho relacionados. Subdivide-se em produtores diretos (pessoas e ferramentas) e produtores indiretos (organização, equipe e papel). Além disso, os produtores devem cumprir suas responsabilidades pelo desempenho das suas funções e colaborando com outros produtores.

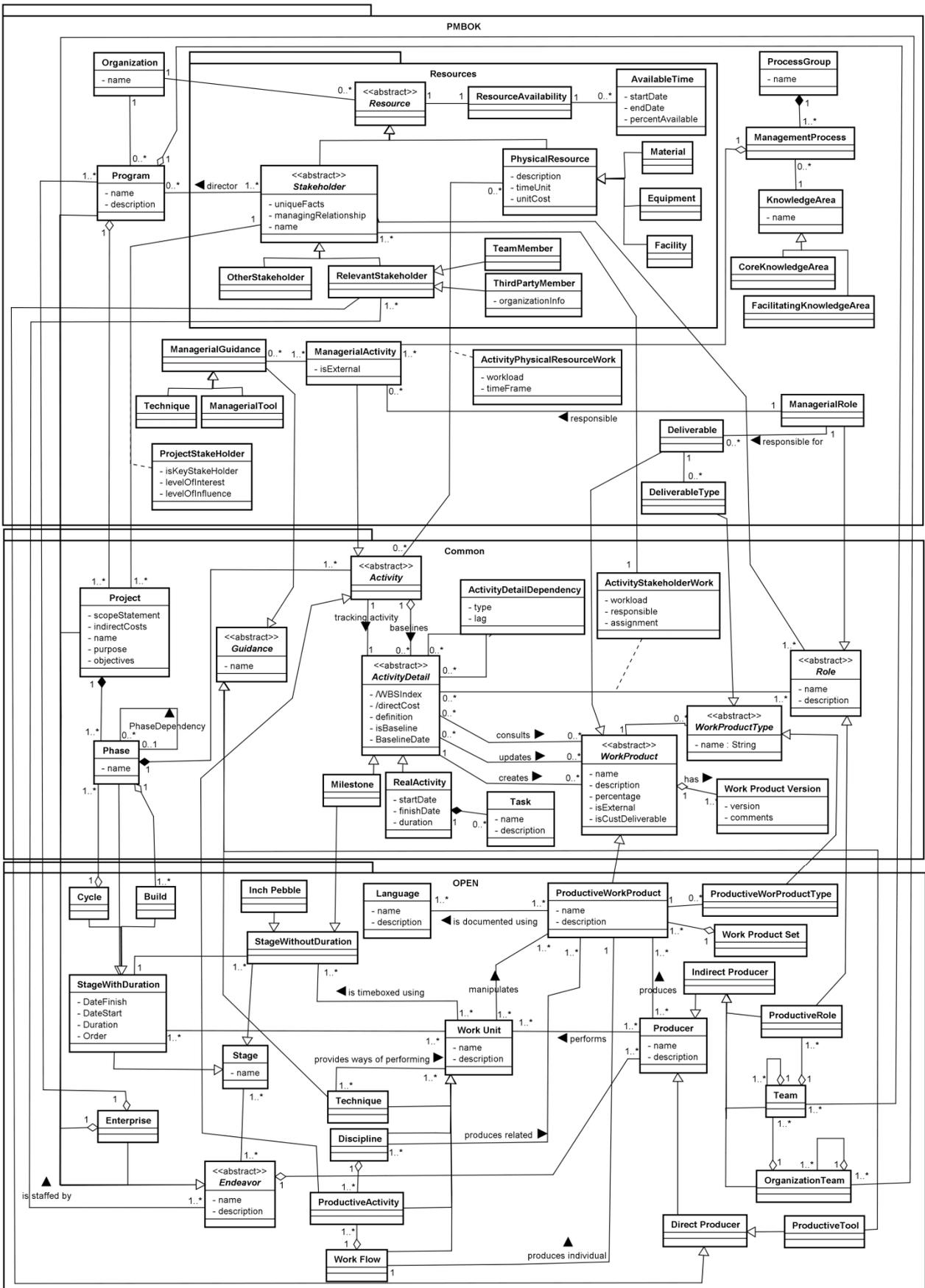



Figura 12. Metamodelo de integração PMBOK+OPEN [ROS12b]

Segundo Firesmith e Henderson-Sellers [FIR02], as unidades de trabalho (classe `WorkUnit`) modelam as operações coesas que são executadas pelos produtores durante o processo de entrega do projeto. Estas são classificadas como tarefas, técnicas, fluxos de trabalho e atividades. A classe `Discipline` representa um conjunto de atividades produtivas que possuem um objetivo comum. A disciplina produz um conjunto de um ou mais produtos de trabalho afins. A classe `Workflow` é constituída por um conjunto de atividades produtivas que produz um único produto de trabalho ou fornece um único serviço. A classe `Technique` é responsável pela modelagem formas de executar uma ou mais unidades de trabalho. Finalmente, a classe `ProductiveActivity` descreve as atividades produtivas realizadas por papéis produtivos.

A classe `Stage` representa os intervalos de tempo que fornecem uma organização macro às unidades de trabalho, sendo subdividida em estágios com duração (ciclos, fases, fluxos, projeto, desenvolvimento, versões de entregas e entregas) e *milestones*.Esta classe é subdivida em etapas com duração (classe `StageWithDuration`), tais como ciclos, fases e *builds*; e estágios sem duração (classe `StageWithoutDuration`), como marcos e *inch-pebbles*. A classe `Cycle`representa um período de tempo em que uma ou mais unidades de trabalho pode ser executada. Um ciclo consiste em uma ou mais fases. A classe `Build` é responsável pela decomposição das fases em períodos gerenciáveis de tempo. Estes períodos de tempo devem ter uma duração curta (por exemplo, um dia ou um mês). A classe `InchPebble` representa marcos em miniatura.

A classe `WorkProduct` representa algo que é produzido, consumido ou modificado (como documentos, modelos ou códigos-fonte), durante a execução das atividades. Um produto de trabalho pode ser subdividido em produtos gerenciais (classe `Deliverable`) ou produtos produtivos (classe `ProdutiveWorkProduct`). A classe abstrata `WorkProductType` contém informações sobre o tipo de um produto de trabalho em um projeto de software específico. Assim, a classe `DeliverableType` descreve uma categoria de produto de trabalho gerencial, tais como atas de reunião, e a classe `ProdutiveWorkProduct` descreve uma categoria de produto de trabalho produtivo, como o modelo UML ou biblioteca de códigos. A classe `ProductiveWorkProduct` (originalmente denominada `WorkProduct` no metamodelo OPF) representa um produto

do trabalho que é produzido, consumido ou modificado durante a execução de atividades produtivas por papéis produtivos.

O conjunto de produtos produzidos pelas tarefas de uma ou mais atividades é representado pela classe `Work Product Set`, enquanto que a classe `Work ProductVersion` corresponde a uma versão específica do produto obtido através do processo de desenvolvimento incremental e iterativo. Ainda, classe `Language` representa as linguagens utilizadas para documentar produtos de trabalho.

3.3.3 Modelo Integrado para o PMBOK e o SPEM

O metamodelo do SPEM 2.0, por sua vez, separa a engenharia dos processos de desenvolvimento de software em dois momentos principais: a criação de um repositório de conteúdo do processo (*Method Content*) e a utilização deste conteúdo (*Process Structure*) em um processo de desenvolvimento de software.

O pacote *Process Structure* do SPEM contém os elementos estruturais básicos para a definição de processos de desenvolvimento de software. Os processos de desenvolvimento de software definem como os projetos de software devem ser executados. Uma das características mais comuns encontradas, dentro das diferentes definições de processo na literatura, é o sequenciamento de fases e marcos expressando um ciclo de vida de um produto em desenvolvimento. Processos também definem como ir de um marco para o próximo através da definição de sequências de trabalho, operações, ou eventos que normalmente ocupam o tempo, perícia, ou outro recurso e que produzem algum resultado. O comportamento dos processos pode ser modelado através da fusão (*merge*) do pacote `Process Structure` com o pacote `Behavior`. As descrições textuais para os elementos dos pacote `ProcessStructure` e `Behavior` podem ser adicionados pelo método de *merge* destes pacotes com o pacote `ManagedContent`.

Conforme este metamodelo, vide Figura 13, a classe `Activity` pertence é uma subclasse de `WorkBreakdownElement` e `WorkDefinition` (do pacote `Core`). Uma atividade define as unidades básicas de trabalho dentro de um processo. Esta classe se relaciona com a classe `WorkProductUse` através da classe `ProcessParameter`. Além disso, a classe `Activity` se relaciona com a classe `ProductiveRole` (originalmente chamada de `RoleUse` pelo SPEM) através da classe `ProcessPerformer`. Uma

atividade suporta o aninhamento e agrupamento lógico dos elementos contidos na estrutura analítica do projeto (subclasses de `BreakdownElements`).

O auto relacionamento definido para o elemento `Activity` chamado `usedActivity` permite o reuso do conteúdo definido para uma atividade em outra atividade. Dessa forma, torna-se possível herdar a estrutura definida para uma atividade em termos de seus elementos aninhados em uma segunda atividade. O metamodelo SPEM 2.0 define alguns tipos de herança para o relacionamento `usedActivity`, os quais são estabelecidos pelo atributo `useKind` da metaclasse `Activity` e pela metaclasse de enumeração `ActivityUseKind`. Esses tipos de herança são os seguintes:

- na: este é o valor default do atributo `useKind` de todas as atividades. Esse valor é usado para atividades que não instanciam a relação `usedActivity`.
- *extension*: este mecanismo permite reutilizar, dinamicamente, as subestruturas (elementos aninhados pela relação de composição) de uma atividade em outras atividades. Dessa forma, a atividade que é associada à outra, pela relação `usedActivity` e possui o valor para o atributo `useKind` igual a extension herda todo conteúdo dessa atividade.
- *localContribution*: este mecanismo permite que adições locais (contribuições) sejam feitas em atividades que estão sendo herdadas através do mecanismo de extensão (*extension*). Uma atividade A poderia, por exemplo, herdar toda estrutura da atividade B pelo mecanismo de extensão (*extension*). Contudo, poderia ser necessário fazer adições locais (contribuições) para a atividade A, através da relação `localContribution`.
- *localReplacement*: este permite que substituições locais sejam feitas em atividades que estão sendo herdadas, através do mecanismo de extensão. Uma atividade A poderia, por exemplo, herdar toda estrutura da atividade B pelo mecanismo de *extension*. Contudo, poderia ser necessário fazer substituições locais para partes da atividade A, através da relação `localReplacement`.

A relação `supressedBreakdownElement` é definida entre as metaclasses `Activity` e `BreakdownElement`. Essa relação permite suprimir elementos de uma atividade após o mecanismo de extensão (*extension*) ter sido realizado. Após uma atividade A, por exemplo, ter herdado todo conteúdo da atividade B, fazendo ou não

contribuições e/ou substituições locais, é possível ainda que elementos sejam suprimidos da atividade A.

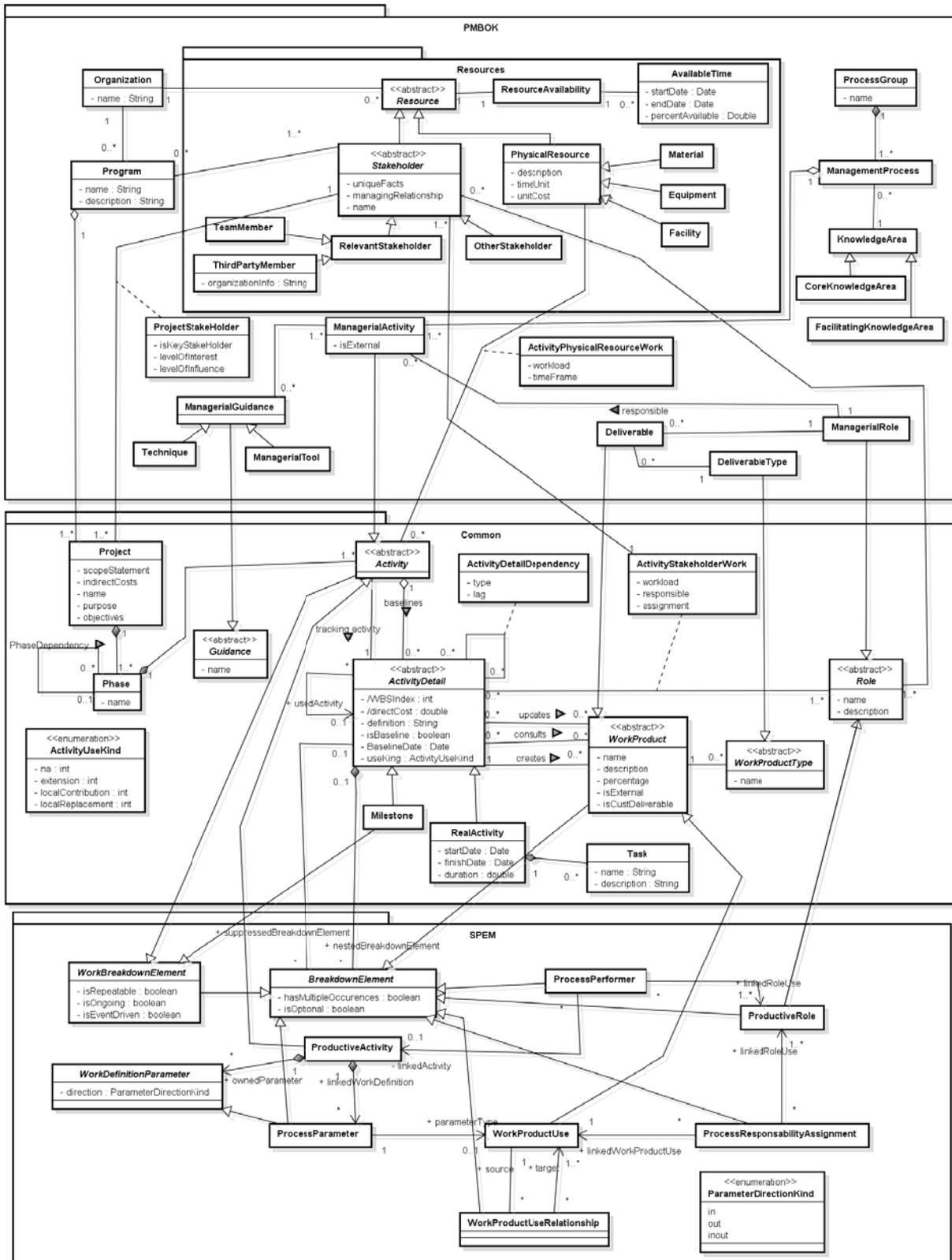



Figura 13. Metamodelo de integração PMBOK+SPEM [ROS12a]

A relação nestedBreakdownElement também é definida entre as metaclasses Activity e BreakdownElement. Ela permite o aninhamento de elementos (tal como outras atividades, marcos, etc.) dentro de uma atividade.

A relação entre a classe Activity e a classe ProcessParameter estabelece, através do atributo direction e da metaclasse de enumeração ParameterDirectionKind, os parâmetros de entrada e/ou saída para as atividades em termos de produtos de trabalho (classe WorkProductUse).

A classe WorkProductUseRelationship estabelece relações (dependência, composição e agregação) entre as classes WorkProductUses. A relação de dependência é citada por vários estudos, tais como [YOO01] [BAJ07] [OMG12] [PAR11], e estabelece que um produto de trabalho (source) depende de um ou mais produtos de trabalho (target) para ser produzido em um processo de software. Já as relações de agregação e composição possuem o mesmo significado de relações da UML e estabelecem que um produto de trabalho (source) pode ser composto por um ou mais produtos de trabalho (target) em um processo de software. A diferença entre as relações de agregação e composição é que, assim como na UML, uma relação de composição estabelece uma relação mais forte entre os produtos de trabalho. Nesse tipo de relação, o(s) produto(s) de trabalho que representa(m) a(s) parte(s) na relação não faz (em) sentido sem o produto de trabalho que representa o todo desta relação.

A classe ProcessPerformerestabelece a relação entre as atividades e os papéis no processo de software. A classe ProcessResponsabilityAssignment estabelece a relação de responsabilidade entre os papéis (classe ProductiveRole) e os produtos de trabalho. Originalmente, no metamodelo SPEM 2.0, é estabelecido que um produto de trabalho pode estar associado a zero ou vários papéis responsáveis em um processo de software, o que é representado com a multiplicidade "0..*" no relacionamento entre as metaclasses WorkProductUse e ProcessResponsabilityAssignment. Exemplos típicos para a classe ProcessResponsabilityAssignment são: Executor Primário, Executores adicionais, Executor Auxiliar, Executor Supervisor, etc.

3.4 Considerações Finaisdo Capítulo

Este Capítulo apresentou os conceitos relacionados aos fluxos organizacionais objetivando salientar a distinção entre as atividades específicas em um projeto e as atividades que fazem parte do fluxo comum de trabalho da organização. Ambos os tipos de fluxos de trabalho são executados em paralelo, possuem recursos próprios e podem influenciar nos prazos das atividades e custos do projeto de software. Sendo assim, o gerente de projetos deve lidar com essa dissociação entre o fluxo de atividades de um projeto de software e os fluxos organizacionais (este último procura oferecer algum tipo de suporte para o projeto de software).

Ainda neste Capítulo, foram apresentados os modelos de referência que foram usados como base para a elaboração do modelo de reconfiguração proposto nesta tese de doutorado. Os modelos de referência foram comparados e combinados em um modelo integrado. Com relação à visão sobre o gerenciamento dos projetos, optou-se por usar como referência o Guia do PMBOK. Para os processos de desenvolvimento de software, foram pesquisados o RUP, OPEN e SPEM.

Após o entendimento sobre a necessidade de integração dos conceitos advindos da gestão de projetos com os processos de desenvolvimento de software, o próximo Capítulo discorre sobre os resultados do estudo realizado para determinar o estado da arte sobre a reconfiguração dinâmica de projetos de software.

.

4 RECONFIGURAÇÃO DINÂMICA DE PROJETOS

As empresas de software, frequentemente, fazem uso de conhecimentos de gestão de projetos para a construção de suas soluções com qualidade e dentro das restrições de escopo, tempo e recursos. O gerente, geralmente, cria um plano de projeto para especificar e limitar o escopo do projeto, e descreve a estrutura analítica do projeto (em inglês, *work breakdown structure* - *WBS*) e o cronograma do projeto. Segundo Schwalbe [SCH02],a WBS contém o esforço de trabalho envolvido em um projeto e define o escopo total do projeto. Por este motivo, é um dos documentos principais da gestão de projetos, pois fornece a base para o planejamento de cronogramas e para a gestão de custos e recursos.

Ao criar um cronograma do projeto, o gerente começa com um conjunto de tarefas contido na estrutura analítica do projeto [PRE09]. Então, ele especifica todas as informações relacionadas ao projeto, como as tarefas individuais, a sequência em que as tarefas precisam ser realizadas, as tarefas que podem ser executadas em paralelo, e os recursos para executar essas tarefas. Com estas informações o gestor pode gerar gráficos apropriados, tais como o gráfico de Gantt, para refletir a alocação real de recursos no projeto. Durante o tempo de vida de um projeto, dados reais, como o tempo ou os recursos que foram gastos para realizar uma determinada tarefa são coletados e inseridos pelo gerente de projeto. Esta informação faz com que seja possível controlar o andamento do projeto e tomar as medidas apropriadas, como a alocação de mais recursos, se necessário.

Durante o planejamento e execução de projetos de software, os gestores também devem levar em conta os diferentes tipos de tarefas que são atribuídos a recursos com características diferentes. Em resposta às novas informações, os gestores podem precisar realizar alterações no plano de projeto, tais como a realocação de recursos ou cancelamento de tarefas [JOS05]. Esses ajustes, necessários para os projetos de acordo com as modificações que ocorrem durante seu ciclo de vida, são denominados nesta tese como sendoparte da reconfiguração dinâmica de projetos.

Este capítulo apresenta uma introdução à área de estudo e também traz algumas definições assumidas no contexto desta pesquisa. Em seguida, são apresentados os principais conceitos sobre a reconfiguração dinâmica no contexto de projetos de software.

4.1 Programação Dinâmica deSistemas

A maioria dos sistemas, inclusive os de software, opera em ambientes dinâmicos onde ocorrem, geralmente, imprevisíveis e inevitáveis eventos em tempo real. Estes eventos podem causar uma mudança na programação dos planos de projeto de modo que um cronograma previamente viável pode se transformar em inviável durante o desenvolvimento do projeto. Exemplos de tais eventos em tempo real incluem falhas de hardware, mau funcionamento de softwares, alterações de data de entrega, doenças de funcionários, entre outros. MacCarthy e Liu [MAC93] abordaram a natureza da lacuna entre a teoria e a prática de sobre a programação de atividades, o fracasso da teoria da programação clássica para responder às necessidades de ambientes práticos e ágeis, e as tendências nas pesquisassobre a programação de atividades que procuram torná-la mais relevante e aplicável nas empresas. Shukla e Chen [SHU96], em sua pesquisa sobre o controle inteligente e em tempo real de sistemas flexíveis de manufatura, afirmaram que existe pouca correspondência entre teoria e prática em relação à programação de atividades. Cowling and Johansson [COW02] abordaram sobre a importante lacuna entre teoria e prática da programação de atividades e afirmaram que os modelos de programação e os algoritmos atuais são incapazes de fazer uso de informações em tempo real.

O problema da programação das atividades de projetos considerando a presença de eventos em tempo real, denominada de programação dinâmica, é de grande importância para o sucesso da implementação de sistemas de softwares nas empresas. No entanto, poucos estudos têm sido publicados nesta área. Esta pesquisa preocupa-se com a questão de como lidar com a ocorrência de eventos em tempo real durante a execução de projetos de software.

4.1.1 O Problema da Programação Dinâmica

A literatura sobre programação dinâmica tem considerado um número significativo de eventos em tempo real e seus efeitos. Estes conceitos, embora sejam comumente utilizados em indústrias, também podem ser aplicadas em organizações de software.Assim, os eventos em tempo real foram classificados em duas categorias [STO96] [SUR93] [COW03] [VIE03]:

- Relacionados a recursos: quebra de hardware, doença de integrante do projeto, indisponibilidade ou falha de software, atraso na chegada ou escassez de

materiais, equipamento defeituoso (equipamento com especificação errada), etc.

- Relacionados ao trabalho: mudança na prioridade da tarefa, cancelamento de tarefas, devido a alterações de data das tarefas, etc.

Nesta pesquisa, a programação dinâmica foi definida em três categorias [MEH99] [VIE00a] [AYT05] [HER05]: programação completamente reativa, programação preditiva-reativa, e programação pró-ativa robusta.

4.1.1.1 Programação Completamente Reativa

Na programação completamente reativa nenhuma programação do projeto é gerada com antecedência e as decisões são tomadas em tempo real. Neste modelo são adotadas regras relacionadas à priorização de atividades. A prioridade de uma atividade é determinada com base em diferentes variáveis: modelo de desenvolvimento de software, tamanho da equipe, restrições relacionadas ao tempo e ao custo, etc. A implementação da programação reativa é, geralmente, intuitiva e fácil de se implementar em projetos de software. No entanto, neste modelo pode ser difícil para o gerente prever o desempenho global do projeto, uma vez que as decisões são tomadas somente em tempo real.

4.1.1.2 Programação Preditiva-Reativa

A programação preditiva-reativa é a abordagem de programação dinâmica de atividades mais utilizada em sistemas de software. A maioria das definições descritas na literatura sobre programação dinâmica refere-se à programação preditiva-reativa. Este é um processo de programação e reprogramação do projeto em que as atividades são revistas em resposta a eventos ocorridos em tempo real. O modelo de programação preditiva-reativa foi adotado nesta tese para o desenvolvimento da solução para a reconfiguração dinâmica de projetos de software.

A maioria das estratégias de programação preditiva-reativa são baseadas em ajustes simples na programação, que consideram apenas a eficiência do projeto. A nova programação pode divergir significativamente da programação original, o que pode afetar seriamente outras atividades que foram planejadas baseadas no cronograma original e, consequentemente, pode levar a fraco desempenho da programação. Consequentemente, é desejável gerar programações preditiva-reativa robustas. A programação preditiva-reativa robusta se concentra na construção cronogramas

preditivos-reativos para minimizar os efeitos de possíveis perturbações sobre o desempenho da programação realizada [WUS91] [WUS93] [LEO94].

Uma solução típica para gerar uma programação robusta é reprogramar as atividades considerando, simultaneamente, a eficiência do projeto e o desvio da programação original (estabilidade). A estabilidade mede o desvio da previsão original da programação das atividades causado pela revisão do cronograma para quantificar a inconveniência de se fazer alterações na programação inicial [WUS91] [WUS93] [COW02] [LEU05]. Wu et al. [WUS91] [WUS93] definiu uma medida de robustez baseada em um conjunto de critérios relacionados ao problema de reescalonamento de atividades considerando um recurso (máquina) com avaria. Estes critérios incluem a minimização do tempo total de trabalho (eficiência da programação) e o impacto da mudança do cronograma (estabilidade da programação). Para a estabilidade, eles investigaram duas medidas: o desvio de tempo de início original do trabalho, e o desvio da sequência original das atividades. Os resultados experimentais mostraram a eficácia desta medida proposta devido ao fato de que a estabilidade da programação foi aumentada de forma significativa, com pouca ou nenhuma redução no tempo total de trabalho. Na mesma linha de raciocínio, Abumaizar e Svestka [ABU97] usaram duas medidas para definir uma programação robusta: eficiência (*makespan*) e estabilidade (desvio do tempo inicial do trabalho e desvios de sequência dos trabalhos). O objetivo da programação é de maximizar a eficiência do projeto e ao mesmo tempo minimizar o impacto no projeto causado por alterações na programação. Jensen [JEN01] investigou diferentes medidas de robustez para melhorar a lentidão e o fluxo total de tempo para quebra de equipamentos (máquinas). Assim, extensos resultados computacionais já relataram a eficiência e a eficácia das medidas robustez propostas acima. Cowling e Johansson [COW02] e Ouelhadj et al. [OUE03] definiram medidas gerais de utilidade e estabilidade para orientar a decisão de qual estratégia deve ser utilizada para reagir a eventos em tempo real, a fim de definir uma programação robusta. A utilidade mede a variação da programação após uma revisão do cronograma. Ela é expressa pela diferença entre a programação das atividades após reagir aos acontecimentos em tempo real e previsão da programação das atividades desenvolvida antes de levar em conta o acontecimento destes eventos.

4.1.1.3 Programação Pró-Ativa Robusta

As abordagens de programação pró-ativa robusta concentram-se na construção de cronogramas preditivos que satisfaçam os requisitos de desempenho previsível em um ambiente dinâmico [MEH99] [VIE03]. A principal dificuldade desta abordagem consiste na determinação das medidas de previsibilidade. Mehta e Uzsoy [MEH99] propôs um modelo de programação previsível com o objetivo de minimizar o atraso máximo.

O efeito da perturbação na programação é medido pelo desvio do tempo de conclusão da tarefa realizada em comparação ao tempo de conclusão planejado. O desvio é reduzido pela inserção um tempo adicional na programação prevista com o objetivo de alcançar alta previsibilidade. Extensivos experimentos computacionais mostraram que a programação previsível proporciona uma melhoria significativa na previsibilidade em consequência de pouca degradação do atraso máximo do projeto. O'Donovan et ai. [ODO99] ampliou a abordagem de escalonamento previsível de Mehta e Uzsoy onde a medida de desempenho do cronograma é o atraso dos trabalhos.

4.1.2 Reprogramação das Atividades na Presença de Eventos em Tempo Real

A reprogramação das atividades do projeto na presença de eventos em tempo real aborda duas questões: como e quando reagir a eventos em tempo real. A primeira questão diz respeito à definição de estratégias de reescalonamento para reagir a eventos em tempo real, e a segunda questão aborda o problema de quando reagendar.

4.1.2.1 Estratégias de Reprogramação das Atividades

Em relação à primeira questão, quais estratégias utilizar para reprogramar as atividades do projeto, a literatura fornece duas estratégias principais de reescalonamento [SAB00] [COW02] [VIE03]: o reparo/ajuste da programação e o reescalonamento completo.

O reparo na programação se refere a algum pequeno ajuste da programação atual. A reprogramação completa, entretanto, gera uma nova programação das atividades a partir do zero. A reprogramação completa pode, em princípio, gerar as melhores soluções, mas essas soluções são raramente possíveis serem implementadas na prática, uma vez que existem diversas variáveis envolvidas. Além disso, a reprogramação completa pode resultar em instabilidade e falta de continuidade do cronograma, levando a produção adicional de custos imputáveis ao nervosismo da equipe.

Sun e Xue [SUN01], e Dorn et al. [DOR95] relataram que a maioria dos sistemas de programação reativa tentar revisar somente parte do cronograma originalmente criado para responder às mudanças no ambiente de produção sem realizar o reescalonamento a partir do zero. Abumaizar e Svestka [ABU97] afirmaram que, na prática, o reescalonamento tem sido feito por meio do reparo da programação, enquanto que a reprogramação completa tem sido usada também, mas de forma limitada. Sabuncuoglu e Bayiz [SA00] demonstraram a eficácia potencial da reparação da programação em termos de estabilidade em comparação com o reescalonamento completo.

Outro problema de importância prática em projetos é a decisão de reprogramar a partir do zero (reescalonamento completo) ou a partir da reparação da programação, e qual a estratégia de reparo da programação deve-se escolher para reagir aos eventos em tempo real. Para lidar com esse problema, foram utilizadas medidas de simulação e robustez para avaliar o desempenho das estratégias de reescalonamento e selecionar a melhor estratégia. Wu et al. [WUS91] [WUS93], Daniels e Kouvelis [DAN95], Abumaizar e Svestka [ABU97] e Jensen [JEN01] utilizaram medidas robustez (eficiência e estabilidade) para decidir sobre a melhor estratégia de reescalonamento aplicar. Cowling e Johansson [COW02], e Ouelhadj et al. [OUE03] utilizaram medidas de utilidade e estabilidade para avaliar o desempenho de vários reparos cronograma e estratégias de reescalonamento completas, e selecionar a melhor estratégia de reprogramação. Com base nestas informações, utiliza-se nesta tese a estratégia de reparo/ajuste da programação do projeto.

4.1.2.2 Quando Reprogramar as Atividades do Projeto

Quanto a segunda questão, quando reprogramar as atividades do projeto, três políticas são propostas na literatura [SAB00] [VIE03]: periódico, dirigido a eventos, e híbrido. As políticas periódicas e híbridas têm recebido atenção especial sob o nome de *rolling time horizon* [CHU92] [OVA94] [SAB99] [VIE00a] [AYT05].

Na política de reprogramação periódica, os agendamentos são gerados em intervalos regulares, que reúnem todas as informações disponíveis do projeto até o momento. O problema da programação dinâmica é decomposto em uma série de problemas estáticos que podem ser resolvidos por meio de algoritmos de agendamento. A programação das atividades será executada, mas não será revisada até que o próximo período inicie. A política de reprogramação periódica pode gerar maior estabilidade e

menor nervosismo da programação. Entretanto, a determinação do período de reprogramação é uma tarefa difícil.

Um importante exemplo sobre a aplicação da abordagem *rolling time horizon* para a programação dinâmica é descrito em [MUH82]. Eles investigaram como a frequência de agendamentos em um ambiente dinâmico de trabalho afetou o desempenho do projeto onde as variações de tempo de processamento e quebra de recursos ocorrem aleatoriamente. Em cada período de reescalonamento, uma programação estática é gerada para os trabalhos atuais. Como antecipado, o desempenho deteriora-se geralmente quando o período de reescalonamento aumenta.

A política de reescalonamento orientada a eventos é acionada em resposta a um acontecimento inesperado que altera o status atual do projeto. A maioria das abordagens de programação dinâmica utiliza esta política. Nesta tese, a solução proposta também se baseia na política de reescalonamento orientada a eventos. Vieira et al. [VIE00b], mostrou em sua pesquisa que a frequência de reescalonamento pode afetar significativamente o desempenho do projeto. Quanto menor for a frequência de reescalonamento, menor será o número ajustes na configuração do projeto. Uma frequência de reescalonamento maior permite ao projeto reagir mais rapidamente às perturbações, mas pode aumentar o número de ajustes.

A política híbrida faz o reescalonamento do projeto periodicamente e também quando ocorre uma exceção. Os eventos normalmente considerados são: avarias de hardware, cancelamento de atividades ou mudanças de prioridade de atividades.

Church e Uzsoy [CHU92] desenvolveram uma política de reescalonamento híbrida de reprogramação onde o projeto não é reescalonado periodicamente. Eventos classificados como sendo de ocorrência regular são ignorados até o momento seguinte de reescalonamento. No entanto, quando um evento é classificado como urgente, a reprogramação completa é executada imediatamente. Os resultados indicam que o desempenho de agendamento periódico deteriora quando o período de reescalonamento aumenta.

4.1.2.3 Técnicas de Programação Dinâmica

A programação dinâmica das atividades dos projetos tem sido foi resolvida utilizando uma diversidade de técnicas [SUH93] [SHU96] [STO96] [BRA99]: heurísticas,

meta-heurísticas, sistemas baseados em conhecimento, lógica fuzzy, redes neurais, técnicas híbridas e os sistemas multiagentes.

4.1.2.3.1 Heurísticas

Heurísticas neste contexto são métodos de reparação da programação para problemas específicos, que não garantem encontrar uma programação ideal, mas tem uma razoável capacidade de encontrar boas soluções em um curto espaço de tempo. As heurísticas mais comuns de reparação programação são: reparação da programação de deslocamento para a direita (*right-shift*), reparação da programação *match-up* e reparação parcial da programação.

A heurística de deslocamento para a direita desloca o tempo das tarefas restantes da programação para frente, considerando a quantidade de tempo ocioso dos recursos envolvidos. A estratégia de reparação da programação *match-up* faz o reagendamento das atividades para possam corresponder com o calendário inicial em algum momento no futuro. A reparação parcial da programação faz o reagendamento apenas das tarefas em atraso ou fracasso.

Abumaizar e Svestka [ABU97] comparam o desempenho da reparação parcial da programação, da reprogramação completa e reparação da programação de deslocamento para a direita (*right-shift*) em relação a medidas de eficiência (*makespan*) e estabilidade (desvio da programação inicial). A heurística da reparação parcial da programação reagenda apenas as tarefas diretamente e indiretamente afetadas pela perturbação causada por algum evento de modo a minimizar, tanto o aumento da eficiência quanto do desvio da programação inicial. Os resultados demonstraram que a heurística reduz o desvio e a complexidade computacional associada à reprogramação completa e ao deslocamento para a direita. O deslocamento para a direita teve o pior desempenho relacionado à eficiência devido ao fato de que o método é um simples deslocamento da programação pelo montante da perturbação causado no cronograma. Assim, quanto mais tempo resultar esta perturbação, maior será o desvio esperado, e maior será o impacto na eficiência da programação.

4.1.2.3.2 Meta-heurísticas:Busca Tabu, *Simulated Annealing* e Algoritmos Genéticos

Nos últimos anos, meta-heurísticas (busca tabu, *simulated annealing* e algoritmos genéticos) têm sido utilizados com sucesso para resolver problemas de programação de

atividades. Metaheurísticas são heurísticas de alto nível que orientam a heurística de busca local [GLO97] [PHA00]. Heurísticas de busca local são métodos de busca na vizinhança baseadas na idéia de procurar seus vizinhos. Em busca de vizinhança local, a busca começa a partir de uma determinada solução, e tenta de forma iterativa avançar para uma solução melhor em uma determinada vizinhança da solução atual usando operadores que se movem. O processo de busca encerra quando não há melhor solução para ser encontrada na vizinhança da solução atual, de modo que este é considerado o local ideal ou ótimo. Meta-heurísticas como a busca tabu, *simulated annealing* e algoritmos genéticos aprimoraram a busca local para escapar dos locais ideais ao aceitar soluções piores, ou através da geração de boas soluções de partida para a busca local de uma forma mais inteligente do que apenas fornecer soluções iniciais aleatórias.

Busca tabu, *simulated annealing* e algoritmos genéticos têm sido amplamente utilizados para resolver problemas determinísticos estáticos de programação de atividades em vários domínios. No entanto, poucas pesquisas abordam o uso de meta-heurísticas na programação dinâmica.

4.1.2.3.3 Programação Dinâmica baseada em Ambiente Multiagentes

Tradicionalmente, os sistemas de controle de cronogramas desenvolvidos em ambientes empresariais têm sido visto como um processo *top-down* de comando, onde resposta que depende fortemente de modelos centralizados e hierárquicos [PAR96] [GOU98] [SHE99] [BOM00] [SHE01]. Para garantir a consistência dos dados em toda a empresa, os sistemas de agendamento centralizado e hierárquico dependem fortemente de bases de dados centrais.

Sistemas de programação centralizada e hierárquica apresentam uma série de inconvenientes [PAR96] [THA97] [SHE99] [BON00] [BRE01]. A principal desvantagem é a existência de um computador central, que constitui um ponto de gargalo que pode limitar a capacidade da empresa, e é um único ponto de falha.Assim, apesar do fato de que os sistemas centralizados e hierárquicos de programação de atividades pode fornecer uma ótima solução para ambientes onde os distúrbios em tempo real são raros, cada vez mais eles tem se mostrado ineficientes em ambientes altamente dinâmicos. Portanto, a programação centralizada e hierárquica é complexa, difícil de manter e reconfigurar, inflexível, onerosa e lenta para satisfazer as necessidades dos ambientes empresariais atuais.

Existem evidências substanciais de que a abordagem de sistemas multiagentes é uma das mais promissoras para a construção de sistemas controle de cronograma devido à sua natureza autônoma, distribuída e dinâmica, e robustez contra falhas [PAR96] [PAR00] [SHE01] [BRE01]. A principal motivação na concepção destes sistemas é descentralizar o controle do sistema agendamento, reduzindo desse modo a complexidade e os custos, aumentando a flexibilidade e aumentar a tolerância a falhas. Um sistema multiagente é composto por uma rede de solucionadores de problemas que trabalham em conjunto para resolver problemas que estão além de suas capacidades individuais [OHA96].

A utilização de sistemas multiagentes para resolver o problema de programação dinâmica é motivada pelos seguintes pontos chave [PAR96] [PAR00] [SHE99] [COW00] [BRE01] [SHE01]. Estes sistemas são compostos de agentes autônomos ligados a cada entidade de física ou funcional na empresa (tais como recursos, atividades e pessoas). A autonomia local permite que os agentes assumam a responsabilidade de realizar a programação local para uma ou mais entidades e de responder localmente e eficientemente a variações locais, aumentando a solidez e flexibilidade do sistema. Em segundo lugar, estes agentes individuais têm liberdade considerável para responder às condições locais para interagir e cooperar uns com os outros, a fim de alcançar programações globais ideais. Além disso, é possível integrar novos recursos ou remover os já existentes com seus agentes ligados ao da empresa. A solução proposta nesta tese de doutorado está associada à solução proposta na dissertação de Schlösser [SCH10]. Nesta dissertação, a autora apresenta uma arquitetura baseada em sistemas multiagentes em resposta aos eventos dinâmicos dos projetos, onde os agentes componentes da solução utilizam o protocolo de rede de contratos [SMI80] para coordenar o processo de geração de propostas. Esta pesquisa tem como foco a programação das agendas individuais de recursos, em tempo real, para atender às diferentes requisições de tarefas de um projeto. Assim, esta solução considera o uso de *solvers*, cuja função consiste em receber a chamada de propostas, analisá-las e gerar propostas para a realocação de recursos que possibilitem prosseguir a execução do projeto.

Um número crescente de organizações está se voltando para tecnologia de agentes para enfrentar os ambientes complexos e dinâmicos comuns à maioria das

empresas. Duas principais arquiteturas multiagentes para a programação dinâmica podem ser consideradas: arquiteturas autônomas e arquiteturas com mediador.

- Arquiteturas autônomas: agentes que representam entidades da empresa, tais como recursos e pessoas têm a capacidade de gerar suas programações locais, reagem localmente para as mudanças e cooperam diretamente uns com os outros para gerar programações globais ótimas e robustas.
- Arquitetura com mediador: tem uma estrutura básica composta de agentes autônomos locais de cooperação que sejam capazes de negociar uns com os outros a fim de alcançar as metas de produção [GOU98] [SHE99] [BON00] [SHE01]. Esta estrutura de base é estendida com o uso de agentes mediadores para coordenar o comportamento dos agentes locais para fazer uma programação global dinâmica. Os agentes mediadores operam simultaneamente com os agentes locais no processo de tomada de decisão. Eles têm a capacidade de aconselhar, impor ou atualizar as decisões tomadas pelos agentes locais, a fim de satisfazer os objetivos globais da empresa e resolver as situações de conflito.

4.1.2.3.4 Outras Técnicas de Inteligência Artificial

Uma série de problemas de programação dinâmica adotaram técnicas de inteligência artificial, como em sistemas baseados em conhecimento, redes neurais, *case-based reasoning*, lógica *fuzzy*, redes de Petri, entre outras, que são discutidas abaixo.

A motivação básica das abordagens baseadas em conhecimento é que há uma grande variedade de conhecimentos técnicos sobre as ações corretivas em resposta a eventos em tempo real. Os sistemas baseados em conhecimento têm como objetivo capturar o conhecimento ou a experiência de especialistas em um domínio específico, e um mecanismo de inferência é utilizado para tirar conclusões ou recomendações sobre a ação corretiva.

Alguns pesquisadores combinaram sistemas baseados em conhecimento e simulação para decidir sobre as melhores ações corretivas em resposta a eventos em tempo real [BEL96]. Outros sistemas baseados em conhecimento, entretanto, foram desenvolvidos para auxiliar o usuário a reagir de forma interativa a eventos em tempo real [DUT90] [SAR90] [HEC00] [OKA00].

Miyashita e Sycara [MIY95] desenvolveram um *framework* para auxiliar na reparação da programação das atividades utilizando o conceito de raciocínio baseado em casos (do inglês: *casebased reasoning*). Casos representam ações adequadas de reparo. O raciocínio baseado em casos permite capturar e reusar desse conhecimento para reparar situações semelhantes. A programação é reparada de forma incremental, quando necessário, utilizando os casos armazenados no sistema.

Redes neurais, redes de Petri, e lógica *fuzzy* também foram utilizadas para resolver o problema de programação dinâmica. Extensas discussões sobre estas técnicas podem ser encontrados em Suresh e Chaudhuri [SUR93], Szelke e Kerr [SZE94], Zweben e Fox [ZWE94], Kerr e Szelke [KER95], Meziane et al. [MEZ00].

Para obter uma melhor programação de sistemas dinâmicos, alguns pesquisadores desenvolveram sistemas híbridos que combinam diversas técnicas de inteligência artificial. Schmidt [SCH94] usou a lógica *fuzzy* para diagnosticar trabalhos críticos a fim de reprogramá-los. Como resultado, o gerente recebe as informações sobre quais as atividades devem ser reagendadas, agora, em breve ou mais tarde. Garetti e Taisch [GAR95], e Garner e Ridley [GAR94] usaram sistemas baseados em conhecimento e redes neurais na programação reativa. As redes neurais foram usadas para decidir sobre o melhor conjunto de regras para definir quais atividades realizar quando ocorre um evento em tempo real.

4.1.2.4 Comparação entre as Técnicas Pesquisadas

A fim de verificar o valor das várias técnicas propostas, foi verificado que alguns trabalhos foram publicados comparando algumas destas técnicas. Estas comparações ajudaram na identificação de quais técnicas são mais adequadas para a programação dinâmica.

As heurísticas têm sido amplamente utilizadas para reagir à presença de eventos em tempo real devido a sua simplicidade. Entretanto, não garantem encontrar uma programação ideal, mas tem uma razoável capacidade de encontrar boas soluções em um curto espaço de tempo. Para superar isso, têm sido propostas meta-heurísticas como a busca tabu, *simulated annealing* e algoritmos genéticos.

Ao contrário da *simulated annealing* e da pesquisa tabu, os algoritmos genéticos manipulam uma população de soluções viáveis. Os algoritmos genéticos, entretanto, não

são eficientes para encontrar uma solução perto do ideal em um tempo razoável em comparação com a busca tabu e simulated annealing [GLO95] [JOZ98] [YOU01] [ZHO01].

Sistemas baseados em conhecimento possuem o potencial para automatizar parte do raciocínio humano para executar sistemas de programação de atividades. Em termos de eficácia da sua capacidade de tomada de decisões, entretanto, eles são limitados pela qualidade e integridade do seu conhecimento de domínio.

A lógica *fuzzy* ainda não foi explorada em todo seu potencial. As redes neurais não podem garantir fornecer decisões ótimas, mas a sua capacidade de aprendizagem as torna ideal para sistemas dinâmicos. Uma possível solução para a programação dinâmica talvez seja a integração de técnicas, tais como redes neurais, simulação e sistemas especialistas.

A maioria dos sistemas de programação de atividades são centralizados e hierarquizados. Sistemas centralizados proporcionam uma visão global consistente da situação da empresa. No entanto, a experiência prática tem indicado que esses sistemas tendem a ter problemas com a reação às perturbações advindas de eventos em tempo real. Assim, diferentes pesquisas dizem respeito à utilização de sistemas multiagentes na programação dinâmica. A principal motivação na concepção destes sistemas é descentralizar o controle de sistemas de programação, reduzindo a complexidade, aumentando a flexibilidade e melhorando a tolerância a falhas. Os agentes possuem a capacidade de observar o ambiente, de se comunicar e cooperar uns com os outros. Sua autonomia local permite que aos agentes responder a variações locais localmente, aumentando a robustez e da flexibilidade do sistema.

Neste capítulo, foram apresentados vários métodos de programação dinâmica, incluindo: heurísticas, meta-heurísticas, sistemas baseados em conhecimento, lógica fuzzy, redes neurais, redes de Petri e os sistemas multiagentes. O estudo comparativo apresentou indicativos de que não há uma solução ótima. Entretanto, a integração de conceitos poderá resultar em um sistema com flexibilidade e robustez necessária para a programação dinâmica.

A próxima seção apresenta a definição e principais conceitos relacionados com a reconfiguração dinâmica de projetos de software.

4.2 Reconfiguração Dinâmica de Projetos de Software: Definição e Conceitos Relacionados

4.2.1 Definição e Aspectos Relacionados

De acordo com o *Project Management Institute*[PMI08], um projeto é um empreendimento temporário com o objetivo de produzir um produto ou serviço. Normalmente um projeto é direcionado para alcançar um resultado específico e envolve a implementação coordenada de atividades inter-relacionadas. Mais do que isso, os projetos são planejados, executados e controlados por pessoas, e são restringidos por recursos limitados. Por sua vez, o gerenciamento de projetos é responsável por monitorar o cumprimento dos objetivos do projeto através da aplicação de um conjunto de técnicas e ferramentas [SCH02]. Assim, observa-se que os gerentes de projetos podem necessitar de algum tipo de apoio, geralmente baseado em uma metodologia de gerenciamento de projetos, para lidar com as diferentes responsabilidades, tarefas e variáveis de projeto. Para esse propósito, existem várias propostas na literatura já aplicadas nas empresas de software.

Entretanto, o gerenciamento de projetos de software não é uma tarefa fácil de ser realizada. Projetos de software são muito dinâmicos e requerem recorrentes ajustes dos seus planos de projeto. Essas configurações podem ser entendidas como reconfigurações na programação, na alocação de recursos e de outros elementos de projeto. Um plano de projeto de software especifica e limita o escopo do projeto, descreve os possíveis riscos do projeto, define os recursos de hardware e software disponíveis, descreve a estrutura de divisão de trabalho e o cronograma do projeto [LEE06]. De acordo com Sommerville [SOM06], o cronograma do projeto especifica relações de dependência, o tempo estimado exigido para alcançar cada etapa do projeto e a alocação das pessoas envolvidas em cada atividade.

Neste cenário, uma configuração de projeto envolve não somente o planejamento das atividades, mas também o planejamento dos recursos disponíveis, suas características e como esses recursos são alocados às atividades. As informações em nível de projeto (como a prioridade sobre outros projetos) e sobre a disponibilidade dos recursos são adicionados ao plano. No entanto, devem-se considerar as mudanças no plano de projeto, especialmente aquelas que ocorrem durante a execução do projeto. Assim, a reconfiguração dinâmica dos projetos envolve sucessivos replanejamentos para

o mesmo projeto durante sua execução. O planejamento de um projeto também envolve a seleção e a definição das atividades necessárias para o cronograma do projeto. Desta forma, o sequenciamento e a dependência das atividades em um projeto determinam a ordem em que estas serão realizadas e, também, quais atividades podem ser executadas em paralelo.

Embora projetos passem por uma fase de planejamento e nessa fase tenha-se conhecimento de objetivos a serem atingidos [PMI08], nem sempre é possível prever o que irá ocorrer durante a fase de desenvolvimento. De acordo com Söderholm [SOD08], para que os projetos sejam desenvolvidos com sucesso, deve ser possível gerenciar eventos não previstos em paralelo com o plano inicial. Uma vez que projetos de desenvolvimento de software geralmente são desenvolvidos em um ambiente com multiprojetos, a ocorrência de um evento pode impactar diversos projetos simultaneamente.

A seguir, são apresentados os eventos identificados que demandam a reconfiguração dinâmica dos projetos de software.

4.2.2 Eventos que Demandam Reconfiguração Dinâmica dos Projetos de Software

À medida que um projeto avança no tempo, determinadas mudanças no ambiente ocorrem e muitas delas podem afetar o planejamento atual do projeto [YU4] [SOD08]. Exemplos frequentes são o atraso de uma tarefa em andamento, mudanças de prioridades e a modificação do grupo de pessoas que formam a equipe [MCC96] [KER06] [LEA04]. Como forma de unificar a nomenclatura, nesta tese tais acontecimentos serão chamados de eventos. No caso específico de software, devido a diversidade de métodos e tecnologias existentes, os quais requerem recursos com conhecimento especializado para o desenvolvimento das tarefas, os eventos podem causar impactos significativos sobre a alocação de diferentes recursos [TSA03].

No caso de cenários com múltiplos projetos, o impacto poderá ser ainda maior. Em sua pesquisa, Engwall e Jerbrant [ENG03] analisaram a chamada “síndrome da alocação de recursos” em um contexto de múltiplos projetos e constataram que os projetos se caracterizam por inúmeras interdependências justamente porque dependem dos mesmos recursos. Assim, identificou-se o problema de definição de prioridades e, consequentemente, de alocação desses recursos.

Para se desenvolver o modelo de reconfiguração proposto neste documento, foi necessário identificar quais eventos perturbam o andamento dos projetos. Os eventos que demandam a reconfiguração dinâmica de projetos de software foram extraídos da dissertação de mestrado de Schlösser [SCH10] e da tese de doutorado de Callegari [CAL10]. Estes trabalhos realizaram uma revisão bibliográfica para a identificação dos eventos, conduzida através de uma revisão sistemática. Após, aplicaram um *survey*com gerentes de projetos de software, com o objetivo de avaliar a frequência e o impacto dos eventos identificados. A combinação dos valores relativos à frequência e ao impacto proporcionou estabelecer uma ordem de prioridade para atacar os tipos de evento.

Cabe salientar que os motivos originais da ocorrência dos eventos não são considerados relevantes para a pesquisa. Para exemplificar, em um evento como a saída de um recurso do projeto, desconsideram-se os reais motivos que o ocasionaram (tais como: demissão, inadequação, realocação do recurso ou qualquer outro motivo). Para esta tese de doutorado, interessa apenas que um evento ocorreu e que devem ser tomadas ações para tratá-lo.

4.2.2.1 Identificação dos Eventos

Para facilitar a identificação, os eventos foram classificados em três categorias:

- Categoria 1: eventos relacionados aos recursos humanos dos projetos de desenvolvimento de software;
- Categoria 2: eventos relacionados às atividades que compõem os projetos de desenvolvimento de software;
- Categoria 3: eventos relacionados a variáveis globais de projetos de desenvolvimento de software.

A Tabela 2 lista os eventos que estão relacionados aos recursos humanos, juntamente com as referências de onde foram extraídos.

Tabela 2: Eventos relacionados aos recursos humanos

ID	Eventos Principais	Referências
ER1	Transferência de um recurso de um projeto A para outro projeto B.	[BEN07]
ER2	Entrada de um novo recurso em um projeto.	[SOD08] [TSA03]
ER3	Saída definitiva de um recurso de um projeto.	[SOD08]
ER4	Suspensão (ou ausência) temporária de um recurso.	[MIL06]
ER5	Alteração de habilidades/competências de um recurso.	[SCH01]
ER6	Alteração dos papéis de um recurso.	[JIA06]

Sobre os eventos relacionados aos recursos humanos, percebe-se que o eventoER1 pode ser ele próprio desmembrado nos eventos ER2 e ER3, uma vez que sair de um projeto e entrar em outro se caracteriza como a própria definição de transferência de recursos entre projetos. A suspensão temporária de um recurso (evento ER4) pode ocorrer por motivos como período de férias, alguma doença ou outro motivo similar.

O evento ER5 refere-se às situações em que um recurso modifica as suas habilidades ou competências. Isso tipicamente ocorre quando o recurso recebeu algum treinamento ou, ainda, quando se verifica que seu conhecimento é insuficiente para uma dada tarefa (em consequência da diminuição de algum grau de suas características). Alteração dos papéis de um recurso pode ocorrem em projetos onde a equipe pode desempenhar mais de um papel (por exemplo, quando uma pessoa desempenha os papéis de analista e desenvolvedor em um mesmo projeto).

Na Tabela 3, encontram-se os eventos que se referem às atividades dos projetos. Os eventos EA1 e EA2 são os mais típicos de qualquer espécie de projeto, pois envolvem estimativas de tempo.A especificação de um novo prazo ou, ainda, a alteração da composição dos recursos, dos papéis ou das competências necessárias foram mapeadas como o evento EA3. Por fim, remover uma atividade ou acrescentar uma nova atividade a um projeto são eventos distintos e foram identificados como EA4 e EA5, respectivamente. Ambos os eventos modificam a rede de atividades do projeto (incluindo as dependências entre as atividades). Dessa forma, todo o subconjunto de atividades afetadas por essa alteração deve ser reconfigurado.

Tabela 3: Eventos relacionados às atividades dos projetos

ID	Eventos Principais	Referências
EA1	Atraso na execução de uma atividade do projeto.	[SCH01], [PMI08]
EA2	Adiantamento na execução de uma atividade do projeto.	[LEA04]
EA3	Alteração de uma atividade (em prazo, recursos, papeis e/ou competências necessárias).	[LEA04], [PMI08]
EA4	Remoção de uma atividade de um projeto.	[MCC96], [LEA04]
EA5	Acréscimo de uma atividade a um projeto.	[MCC96], [LEA04]

ATabela 4 apresenta os eventos que estão relacionados às variáveis globais dos projetos, ou seja, que não são específicos para uma atividade ou para um recurso.

Tabela 4: Eventos relacionados a características globais dos projetos

ID	Eventos Principais	Referências
EP1	Mudanças no escopo do produto (diminuição, aumento ou alteração técnica).	[SOD08], [VER05]
EP2	Alteração no custo limite do projeto.	[SOD08]
EP3	Alteração de milestones do projeto (datas intermediárias e/ou entrega final).	[PMI08]
EP4	Inclusão de um novo projeto no portfólio atual.	[LYC04], [EVE00]
EP5	Cancelamento de um projeto do portfólio atual.	[LEA04]
EP6	Alteração de prioridade/importância de um projeto, quando comparado a outros em andamento.	[BEN07], [ENG03]

Observa-se que alguns dos eventos estão diretamente relacionados entre si. Para exemplificar, quando um projeto tem o seu escopo alterado (evento EP1), tipicamente ocorrem outras modificações sobre o projeto para de adequá-lo à nova realidade. Assim, efetivamente são provocados outros eventos como consequência, tais como os eventos identificados como EA3, EA4 e EA5. Dessa forma, embora na literatura se tenha identificado explicitamente tal acontecimento como um evento, na prática isso se traduz como a ocorrência de outros eventos diretamente relacionados.

A inclusão e o cancelamento de um projeto, por sua vez, também podem ser vistos, na prática, como alterações no *pool* de recursos e no conjunto de atividades a serem realizadas. Sendo assim, quando se registram as atividades de um novo projeto ou quando um projeto é integralmente removido (e, por consequência, todas as suas atividades também), tais alterações afetam tanto os recursos envolvidos quanto as atividades do projeto.

4.2.2.2 Resposta aos Eventos em Tempo Real

Uma vez identificados os eventos, parte-se para a definição de conjuntos de ações que são executadas sempre que um evento de reconfiguração ocorre. Conforme se verificou, há ações gerenciais que podem ser tomadas para minimizar o impacto de determinados eventos. Tais ações estão geralmente incluídas em planos de riscos [PMI08]. No entanto, essas são geralmente ações de longo prazo, como manter alguma reserva ou prever ações de recuperação nos planos de risco.

O interesse desta tese, contudo, está em ações de curto prazo, que devem ser tomadas imediatamente frente a um evento inesperado, para que se restabeleça o

andamento dos projetos o mais rápido possível, ainda que variáveis de longo prazo possam estar presentes.

4.3 Revisão Sistemática sobre a Reconfiguração Dinâmica de Projetos de Software

Após vários estudos preliminares na literatura para identificação de pesquisas relacionadas foi realizada uma revisão sistemática sobre a reconfiguração dinâmica de projetos de software (um resumo do protocolo de pesquisa pode ser encontrado no Apêndice B). A revisão sistemática é uma metodologia de pesquisa que usa métodos sistemáticos para identificar, selecionar e avaliar criticamente os estudos científicos em um campo específico de investigação. É uma revisão planejada para responder a uma pergunta específica que pode ou não incluir métodos estatísticos. Métodos estatísticos utilizados na análise e síntese dos estudos selecionados são chamados de meta-análise [HIG08].

A revisão sistemática é uma forma de avaliar e interpretar o trabalho relevante para uma questão de pesquisa específica [KIT04], [BIO05]. Resumidamente, em uma revisão sistemática realiza-se uma busca de informações relevantes à pesquisa, definindo “palavras de pesquisa" (*search strings*) a serem executadas em mecanismos de pesquisa, tais como os fornecidos pela IEEE e ACM. Esta informação é geralmente relatada através de artigos em anais de congressos, revistas e relatórios técnicos ou mesmo em livros dedicados a cada área de pesquisa. As vantagens de uma revisão sistemática podem ser resumidas como se segue:

- Agrupar as evidências existentes sobre um determinado assunto na literatura;
- Identificar as eventuais lacunas que podem apontar para futuras pesquisas;
- Fornecer uma visão consistente para propor novas atividades de investigação.

A revisão sistemática fornece um rigor científico a um processo de revisão da literatura e, como consequência, minimiza problemas que podem acontecer durante uma revisão da literatura convencional. As diretrizes para conduzir o processo de revisão sistemática estabelecidas por Biolchini et al. [BIO05] e Kitchenham [KIT04] foram adaptadas de modo a refletir os problemas específicos da pesquisa em Engenharia de Software. Essas diretrizes são compostas por três fases: planejamento da revisão, a realização da revisão, e relato da revisão.

4.3.1 Planejamento da Revisão Sistemática

Primeiramente, uma revisão sistemática deve começar com a definição da questão de pesquisa. Neste trabalho adotamos o termo “pergunta foco” (*Question Focus* - QF), como forma de representar a pergunta da pesquisa (vide ApêndiceD). Para determinar a QF, algumas questões complementares foram utilizadas. As próximas subseções resumem a sequência de passos que foram estabelecidas por Biolchini et al. [BIO05] e Kitchenham [KIT04]. Em seguida, a terceira fase é representada através de análise dos resultados.

4.3.1.1 Escopo da Pesquisa

Considerando o escopo geral da pesquisa, o objetivo é encontrar soluções para as seguintes perguntas (Q_x):

- Q_1: Quais são as abordagens existentes e soluções relacionadas à reconfiguração dinâmica da rede de planejamento dos projetos de software?
- Q_2: Como são preparados os processos de projetos de software para gerenciar múltiplos projetos ao mesmo tempo?
- Q_3: Quais são as metodologias de gerenciamento de projetos que apoiam a integração do fluxo de trabalho de projeto de software com outros fluxos organizacionais?

Estas questões foram usadas para delimitar o escopo que será atendido pelo processo de revisão sistemática estabelecido neste trabalho através da análise e síntese dos estudos selecionados. Com base nas perguntas pré-identificadas, a QF foi definida como se segue:

- QF: "Quais abordagens existentes na literatura permitem a reconfiguração dinâmica das redes de planejamento de projetos de software, considerando vários projetos simultâneos e a sua integração com outros fluxos organizacionais?”.

A QF é essencial para determinar a estrutura da revisão. Se a QF não está bem definida, pode-se comprometer substancialmente o resultado da pesquisa. Todos os passos da análise sistemática foram guiados pela QF, que também foi utilizada como uma forma de julgamento da relevância desta revisão sistemática.

4.3.1.2 Detalhes da Revisão Sistemática

Após a definição da QF, os próximos passos são, respectivamente, a seleção das fontes de pesquisa, a definição de critérios de inclusão e exclusão, e a seleção dos estudos encontrados. Quando esse planejamento for concluído, deve-se avaliar o plano de protocolo da revisão sistemática. Se for aprovado, a extração de informações deve começar (ver Figura 14). Os resultados também devem ser avaliados antes de se realizar a análise final. A fim de manter todas as informações e documentar as decisões dos pesquisadores, deve-se fazer o armazenamento dos resultados ao longo deste processo.

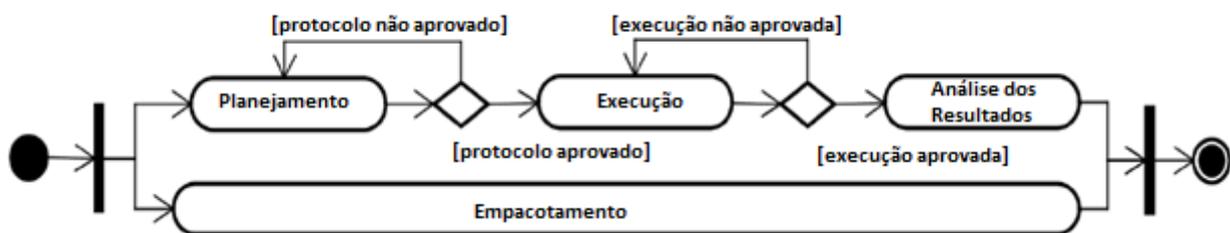



Figura 14. Processo de revisão sistemática – adaptado de [BIO05]

Esta revisão sistemática utilizou os motores de busca a seguir: *IEEE Xplorer digital library, ACM digital library, Springer Link and Science Direct*. Os critérios para tomar a decisão sobre o motor de busca selecionado foram:

(1) O banco de dados permite motores de busca baseados em palavras-chave e expressões booleanas; e

(2) A disponibilidade de artigos através da Internet.

Uma primeira seleção dos trabalhos ocorreu entre março e maio de 2011, cujos resultados foram publicados em [ROS12a]. A população definida para esta revisão inclui artigos publicados em revistas e conferências sobre o campo de estudo desde 2000 até 2011 (período em que foi realizado esta revisão sistemática) e que foram escritos em inglês. O mesmo protocolo desta revisão sistemática foi aplicado novamente em junho de 2012 com o objetivo de captar trabalhos relacionados mais recentes. Este período foi escolhido a fim de realizar um filtro nos trabalhos recentes na área, antes de se aprofundar nesta pesquisa. A justificativa para a escolha do idioma inglês é devido à sua universalidade, mantendo-se a linguagem padrão de conferências e revistas internacionais. Em seguida, vários trabalhos foram analisados nas áreas de ciência da computação, engenharia de software, gerenciamento de projetos, sistemas de informação, sistemas de apoio à decisão e gestão de processos de negócio. As fontes

foram acessadas exclusivamente na Internet, a pesquisa de modo manual não foi considerada nesta tese.

Inicialmente, para a definição da palavra de pesquisa (*search string*), foram julgadas várias combinações e variações das seguintes palavras-chave (em inglês): "*project management*", "*scheduling*", "*businessprocess*", "*workflow*", "*managerialactivity*", "*enterpriseactivity*", "*organizationalactivity*" e "*productiveactivity*". Depois de observar o comportamento dos motores de busca, estas palavras-chave foram combinadas através de filtros e operadores booleanos, conforme apresentado na Tabela 5.

Tabela 5: Strings de pesquisa utilizados

ID	***String* de Pesquisa**
S_1	*(project management OR scheduling) AND (business process OR workflow) AND (managerial activity OR enterprise activity OR organizational activity OR productive activity) AND (year >= 2000 AND year <= 2011)*
S_2	*(project management) AND (business process) AND (managerial activity OR enterprise activity OR organizational activity OR productive activity) AND (year >= 2000 AND year<= 2011)*
S_3	*(scheduling) AND (workflow) AND (managerial activity OR enterprise activity OR organizational activity OR productive activity) AND (year >= 2000 AND year <= 2011)*
S_4	*(dynamic reconfiguration) AND (business process OR workflow) AND (year >= 2000 AND year <= 2011)*

Deve ser observado que algumas *strings* de pesquisa utilizadas retornam um razoável número de documentos. Os 132 artigos selecionados foram resultantes da *string* S_1. Esta *string* seleciona artigos de revistas e conferências, publicados desde 2000, que lidam com a reconfiguração dinâmica dos projetos de software com ênfase na integração da gerência de projetos com os fluxos organizacionais.

4.3.2 Análise dos Resultados

Os 132 artigos foram preliminarmente selecionados para uma análise posterior. Assim, esta seleção foi baseada na seguinte sequência de passos:

- A *string* de pesquisa foi executada em cada um dos motores de busca;
- O título e o resumo de cada artigo foram lidos;
- Quando preliminarmente aprovados, os textos completos foram lidos para uma aprovação final;
- Em caso de dúvida, devido à falta de clareza do resumo do artigo, uma leitura rápida do texto foi realizada;
- Os artigos restantes foram selecionados para uma leitura completa.

A execução desta sequência de passos resultou na seleção de 24 trabalhos. Quatorze deles foram excluídos mais tarde devido à falta de relevância mínima esperada. Assim, a seleção final incluiu 12 artigos (9,09% dos resultados globais dos motores de busca). De acordo com Biolchini et al. [BIO05] e Kitchenham [KIT04], esta redução é esperada em uma revisão sistemática, principalmente devido a erros de seleção dos motores de busca.

O principal critério para seleção dos trabalhos pesquisados foi aceitar apenas aqueles que incluíam um bom nível de descrição sobre a solução e informações suficientes sobre os métodos ou estratégias, de qualquer tipo, para resolver as questões relacionadas com o tema de pesquisa deste trabalho. Mais especificamente, foi procurada a informação sobre os seguintes itens:

1. Atividade de agendamento (*scheduling*): indica se o artigo apresenta uma solução para a programação e agendamento das atividades do cronograma do projeto;
2. Conceito de atividades gerenciais de apoio: indica se o artigo apresenta uma distinção entre atividades específicas do projeto daquelas que são comuns a toda organização;
3. A integração com os fluxos organizacionais: indica se a solução fornece integração do fluxo de atividades do projeto com os fluxos organizacionais;
4. Suporte a multiprojetos: indica se a solução suporta mais de um projeto simultaneamente;
5. Tipo de solução: indica se ele mostra a solução da "categoria geral" (por exemplo, de apoio à decisão, otimização e metodologia.).
6. Método: o método utilizado para a solução (por exemplo, redes Bayesianas e programação dinâmica).
7. Solução dinâmica: indica se a solução pode fornecer *feedback* imediato durante o curso do projeto;
8. Uso de simulação: indica se a solução envolve algum tipo de simulação em computador;
9. Avaliação pela comunidade científica: Indica se a pesquisa foi cientificamente avaliada como um estudo de caso ou experimento;
10. Ferramenta: indica se a solução inclui uma ferramenta ou um protótipo.

A classificação dos estudos com base nestes critérios é apresentada na Tabela 6.

Tabela 6: Resultado sobre os artigos pesquisados

Fonte	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10
[CAL07]	Sim	Sim	Sim	Sim	Suporte à decisão	Modelo de Integração	Sim	Não	Não	Não
[CHE05]	Sim	Não	Não	Não	Suporte à decisão	Modelo Matemático	Sim	Não	Não	Não
[LEE04]	Sim	Não	Não	Sim	Suporte à decisão	Cenários	Sim	Sim	Não	Sim
[JOS05]	Sim	Sim	Sim	Não	Suporte à decisão	Estratégias	Sim	Sim	Não	Não
[RIH01]	Sim	Não	Não	Não	Suporte à decisão	Multiagente	Sim	Sim	Sim	Sim
[ZHA09]	Sim	Não	Não	Sim	*Framework*	*Workflow*	Sim	Não	Não	Sim
[AYE05]	Sim	Não	Não	Não	*Framework*	*Workflow*	Sim	Não	Não	Sim
[DEL99]	Sim	Sim	Não	Sim	Suporte à decisão	*Toolkit*	Sim	Não	Sim	Sim
[DIW08]	Sim	Não	Não	Não	*Framework*	*Workflow*	Sim	Não	Não	Não
[GRU04]	Sim	Sim	Não	Não	Suporte à decisão	Modelo	Sim	Não	Não	Não
[HES09]	Sim	Não	Não	Sim	Suporte à decisão	Algoritmo	Sim	Sim	Não	Não
[HEL03]	Sim	Não	Não	Sim	Suporte à decisão	*Workflow*	Sim	Sim	Não	Não

Através da análise daTabela 6, podem-se obter algumas conclusões com base em uma rápida análise quantitativa:

- Primeiramente, todos os trabalhos selecionados apresentam soluções para a programação e agendamento de atividades do cronograma;
- Quatro dos artigos analisados apresentam uma distinção entre as atividades técnicas e gerenciais;
- Apenas dois artigos apresentam soluções que propiciam a integração das atividades do fluxo do projeto com os fluxos organizacionais da empresa;
- Quatro papéis lidar com cenários multiprojetos;
- É interessante notar que todos os resultados mostram soluções que envolvem algum tipo de solução dinâmica;
- Quatro estudos apresentaram soluções que envolvem algum tipo de técnica de simulação;
- Uma observação importante é que mais da metade dos artigos selecionados têm algum tipo de ferramenta ou protótipo. No entanto, apenas um estudo inclui resultados de um estudo de caso ou experimento.

O planejamento das atividades é fortemente afetado por incertezas e eventos externos [JOS05] [JAL04]. A ideia de usar planos de contingência parece ser muito promissora e combate boa parte da subjetividade presente nas decisões diárias feitas por gerentes de projetos de software. Os planos de contingência são sequências pré-codificadas de ações que devem ser realizadas toda vez que um problema ocorre. Portanto, esta é uma solução dinâmica. Em outras palavras, diante de um problema na configuração atual, é permitida a adoção de estratégias de solução para cada situação apresentada.

O trabalho apresentado por Delen et al.[DEL99] contém um conjunto de ferramentas integradas para modelagem, análise e gestão de sistemas. A ferramenta PROSIM, por exemplo, fornece mecanismos para modelagem, análise e desenho de processos de negócios. Ela fornece um ambiente gráfico para modelar processos de negócio e, em seguida, realizar simulações de cada processo. Outra ferramenta, o ProjectLINK, que é um *add-on* para PROSIM, permite que as informações a partir de um modelo de processo sejam transportados para uma ferramenta de gerenciamento de projetos, por exemplo, o Microsoft Project.

Em [RIH01], é mostrado o ProPlanT, uma ferramenta multiagente que permite o planejamento das atividades de produção e seleção de recursos (com base em mecanismos de subscrever / anunciar). Em [HES09], é apresentado um algoritmo de agendamento de recursos formulado para resolver o problema de atribuição de recursos uma grade de fluxos de trabalho de produção (em inglês, *manufacturing grid workflow*). Em [HEL03], é apresentado um sistema de gestão que foi desenvolvido dar apoio a processos de engenharia de projeto.

Ainda, no âmbito da revisão sistemática, os dois únicos estudos que mostram a integração com os fluxos organizacionais são [CAL07][JOS05]. O primeiro apresenta um modelo de integração do PMBOK com o RUP, conceituando as atividades gerenciais e produtivas e as relações de interdependência entre estes dois tipos de atividades. O segundo somente apresenta um exemplo de contratação temporária de recursos. Junto à ideia de integrar as atividades de um projeto de software com os fluxos de atividades organizacionais, o artigo apresentado por Lee e Miller [LEE04] trabalha com o conceito de política (*policy*) ou políticas. Este trabalho apresenta algumas estratégias que o gestor pode considerar para gerenciar seu projeto, por exemplo, alocando mais recursos para uma atividade, criando marcos internos, entre outros. Ele admite projetos simultâneos,

mas estes podem ter características diferentes (por exemplo, o primeiro projeto pode ser executado com base em um processo iterativo, enquanto que o segundo pode ser baseado no modelo em cascata).

Através desta análise, observa-se que as obras pesquisadas não fornecem uma solução que permita o planejamento de projetos de software, considerando as interações entre o gerente de projeto com outros departamentos dentro da organização. A desconsideração das dependências entre as atividades de diferentes fluxos de trabalho pode resultar em distorções no planejamento de projetos de software. Muitas vezes, o gerente de projeto somente sente a necessidade de solicitar as informações de outros departamentos da empresa ao executar uma determinada atividade do projeto que depende destas informações externas ao projeto (por exemplo, aquisição de equipamentos ou a contratação de um novo desenvolvedor). Consequentemente, existe a necessidade de uma solução que permita antecipar as necessidades resultantes destas áreas de apoio da empresa durante o planeamento de projetos de software. Esta solução deve considerar a complexidade de identificar a interdependência entre as atividades dos fluxos de trabalho da organização e o fluxo de trabalho dos projetos. No entanto, nenhum dos trabalhos retornados nesta revisão sistemática fornece algum mecanismo computacional para resolver todos esses problemas relatados nesta pesquisa.

4.3.3 Limitações da Revisão Sistemática

Conforme observado no Apêndice B, a execução desta revisão sistemática apresentou as seguintes restrições:

- Consideraram-se somente os artigos retornados pelos motores de busca selecionados, ou seja, IEEE Xplorer digital library, ACM digital library, Springer Link and Science Direct.
- Os artigos analisados ficaram restritos àqueles publicados nos idiomas inglês e português;
- Consideraram-se somente as publicações realizadas os anos 2000 e 2011;
- A *string* de pesquisa que foi utilizada na preparação da revisão sistemática pode acabar restringindo alguns artigos retornados pelos motores de busca, de modo que se acaba assumindo o risco de deixar de fora alguns trabalhos relacionados.

- Alguns critérios de inclusão e exclusão de estudos foram adotados na seleção dos estudos. Desta forma, foram desconsiderados os estudos:
    - Que não estavam no formato de artigo completo (*full paper*).
    - Cujo conteúdo não estava relacionado a área de reconfiguração de projetos.
    - Trabalhos que citem importância da reconfiguração de projetos, mas que não possuam nenhum tipo de solução para isto.
- Também foram removidos estudos do tipo *roadmaps* (trabalhos que indicam os desafios e as direções a serem tomadas em uma área de pesquisa)

4.4 Considerações Finaisdo Capítulo

Este Capítulo apresentou uma introdução à área de estudo e também trouxe algumas definições assumidas no contexto desta pesquisa. Em seguida, foram apresentados os principais conceitos sobre a reconfiguração dinâmica no contexto de projetos de software.

Neste Capítulo, também foram apresentados vários métodos de programação dinâmica, incluindo: heurísticas, meta-heurísticas, sistemas baseados em conhecimento, lógica *fuzzy*, redes neurais, redes de Petri e os sistemas multiagentes. O estudo comparativo apresentou indicativos de que não há uma solução ótima.

O próximo Capítulo apresenta o estudo realizado nesta tese sobre a utilização de redes de Petri na gestão de projetos.

5 APLICAÇÃO DAS REDES DE PETRI NA GESTÃO DE PROJETOS

5.1 Vocabulário e Conceitos Associados

A rede de Petri (RdP) é uma ferramenta gráfica capaz de modelar, analisar, controlar, validar e implementar sistemas, em especial aqueles que possam ser interpretados como sistemas a eventos discretos. Sistema discreto é um sistema no qual as mudanças ocorrem em instantes precisos. Os sistemas a eventos discretos possuem estados bem definidos e a mudança de estado acontece quando da ocorrência de um evento [MUR89] [CAR97]. Um evento, do ponto de vista de sistema de software, é uma ocorrência de origem interna ou externa, que altera as características do fluxo do projeto, provocando mudanças de estado do sistema.Na seção seguinte é apresentada a caracterização de tais sistemas e os conceitos básicos utilizados na sua modelagem.

5.1.1 Sistemas Discretos

Um sistema discreto é um sistema onde as mudanças de estado ocorrem em instantes bem definidos. Costuma-se situar os sistemas discretos em oposição aos sistemas contínuos. Esta classificação depende do ponto de vista em que se coloca o observador e depende do grau de abstração desejado. Pode-se, de fato, encontrar diversas definições de tais sistemas, representados na Figura 9, que são enumerados a seguir:

- Sistemas discretizados: são sistemas estudados somente em instantes precisos. Trata-se, portanto, de sistemas contínuos observados em instantes discretos (sistemas amostrados). As variáveis de estado evoluem de maneira contínua, sem mudança brusca de comportamento, mas é somente a instantes discretos do tempo que há interesse em conhecer seu valor.
- Sistemas discretos: são sistemas para os quais os valores das variáveis de estado, ou ao menos de algumas delas, variam bruscamente a certos instantes. Entretanto, estes instantes não podem necessariamente ser previstos e o conhecimento do estado a um instante dado não permite que, sem cálculo, se conheça o estado seguinte.

- Sistemas a eventos discretos: são sistemas modelados de tal sorte que as variáveis de estado variam bruscamente em instantes determinados e que os valores das variáveis nos estados seguintes podem ser calculados diretamente a partir dos valores precedentes e sem ter que considerar o tempo entre estes dois instantes. É esta classe de sistemas que será estudada nesta tese.

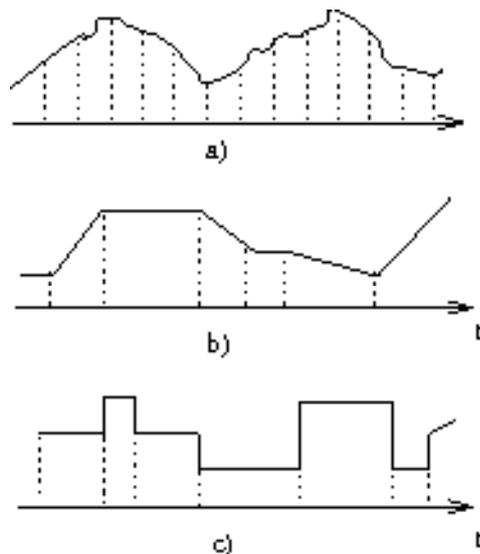



Figura 15. Sistemas: a) discretizado; b) discreto; c) a eventos discretos [CAR97]

5.1.2 Conceitos Utilizados na Modelagem de Sistemas a Eventos Discretos

Os conceitos básicos utilizados na modelagem de um sistema baseado numa abordagem por eventos discretos são os seguintes:

- Eventos: são os instantes de observação e de mudança de estado do sistema.
- Atividades: são as caixas-pretas utilizadas para recuperar e esconder a evolução do sistema físico entre dois eventos. Portanto, os eventos correspondem em geral ao início e ao fim de uma atividade.
- Processos: são sequências de eventos e de atividades interdependentes. Por exemplo, um evento provoca uma atividade, que provoca um evento de fim de atividade, que por sua vez pode provocar outra atividade e assim por diante.

5.1.3 Paralelismo, Cooperação, Competição

A evolução dos processos num sistema pode ocorrer de forma simultânea ou não. Quando ocorrer de forma simultânea, os processos podem ser completamente independentes ou relativamente independentes.Esta independência relativa significa que certas atividades são totalmente independentes entre si, enquanto que outras atividades

necessitam de pontos de sincronização, isto é, de eventos comuns a várias evoluções. De acordo com Murata [MUR89], existem diferentes formas de interação entre processos:

- Cooperação: os processos concorrem a um objetivo comum; procura-se descrever uma independência de processos antes de um ponto de sincronização.
- Competição: os processos devem ter acesso a um dado recurso para realizar sua tarefa. Se existisse um número suficiente de recursos, os processos seriam completamente independentes. Trata-se, portanto, de um partilhamento de recursos resolvido, em geral, por exclusões mútuas. Procura-se, portanto, descrever uma exclusão entre dois processos a partir de um ponto de sincronização.
- Pseudo-paralelismo: o paralelismo é apenas aparente e os eventos, mesmo independentes, nunca serão simultâneos. Eles serão ordenados por um relógio comum. É o caso de várias tarefas sendo executadas num único processador. Este executa somente uma instrução por vez.
- Paralelismo verdadeiro: os eventos podem ocorrer simultaneamente. Isto significa que não existe uma escala de tempo comum suficientemente precisa para determinar qual evento precedeu o outro. Ocorre quando várias tarefas são executadas num computador paralelo, com um processador alocado para cada tarefa independente.

5.2 Introdução às Redes de Petri

A rede de Petri é uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes [MUR89]. Entre as suas aplicações pode-se citar: avaliação de desempenho, análise e verificação formal em sistemas discretos, protocolos de comunicação, controle de oficinas de fabricação, concepção de software em tempo real e/ou distribuído, sistemas de informação, sistemas de transporte, logística, gerenciamento de base de dados, entre outros.

Os elementos básicos que permitem a definição de uma rede de Petri são os seguintes:

- Lugar (representado por um círculo): pode ser interpretado como uma condição, um estado parcial, uma espera, um procedimento, um conjunto de recursos, um

estoque, uma posição geográfica num sistema de transporte, etc. Em geral, todo lugar tem um predicado associado, por exemplo, máquina livre, peça em espera (vide Figura 16);

- Transição (representada por barra ou retângulo): é associada a um evento que ocorre no sistema, como o evento iniciar a operação (transição *t* naFigura 16);
- Ficha (representado por um ponto num lugar): é um indicador significando que a condição associada ao lugar é verificada. Pode representar um objeto (recurso ou peça) numa certa posição geográfica (num determinado estado), ou ainda uma estrutura de dados que se manipula. Por exemplo, uma ficha no lugar "máquina livre" indica que a máquina está livre (predicado verdadeiro). Se não tem fichas neste lugar, o predicado é falso, por conseguinte a máquina não está livre. Se no lugar peças em espera houvesse três fichas, indicaria que existem três peças em espera.

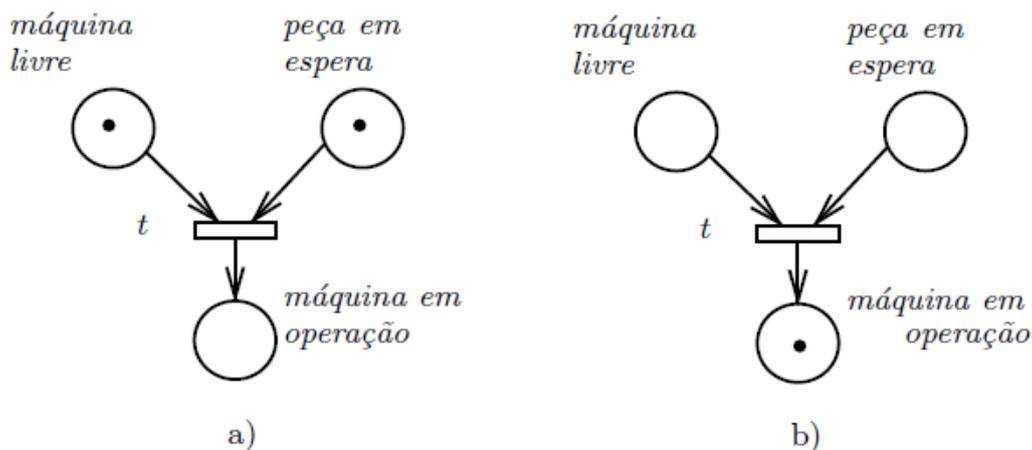



Figura 16. Consumo de fichas pela rede de Petri[CAR97]

O estado do sistema é dado pela repartição de fichas nos lugares da rede de Petri, cada lugar representando um estado parcial do sistema. A cada evento que ocorre no sistema, é associada uma transição no modelo de rede de Petri. A ocorrência de um evento no sistema (que faz com que este passe do estado atual ao próximo estado) é representada, no modelo, pelo disparo da transição ao qual este está associado.O disparo de uma transição consiste em dois passos:

- retirar as fichas dos lugares de entrada, indicando que esta condição não é mais verdadeira após a ocorrência do evento; e
- depositar fichas em cada lugar de saída, indicando que estas atividades estarão, após a ocorrência do evento, sendo executadas.

Por exemplo, a ocorrência do evento “iniciar a operação”, associado à transição *t*(vide Figura 16a), só pode acontecer se houver pelo menos uma ficha no lugar “máquina livre” e pelo menos uma ficha no lugar “peça em espera”. A ocorrência do evento “iniciar a operação”, no sistema, equivale ao disparo da transição *t* na rede de Petri: é retirada umaficha do lugar “máquina livre” e uma ficha do lugar “peça em espera”, e é colocada uma ficha no lugar “máquina em operação” (vide Figura 16b).

O disparo de *t* corresponde à ocorrência do evento *e* no sistema real, que o faz passar de um estado atual E_i ao próximo estado E_{i+1}. O estado E_i é representado na rede pela distribuição de fichas nos lugares, chamada marcação M_i. Do mesmo modo que o sistema só atingirá o estado E_{i+1} após a ocorrência do evento e se estiver no estado E_i, assim também a transição *t* só será disparada se a marcação for M_i (marcação em que, em particular, os lugares de entrada de *t* estão marcados). A marcação M_{i+1}, correspondente ao estado E_{i+1}, será atingida após o disparo da transição *t*.Na Figura 17, são mostrados os efeitos para várias situações onde ocorrem disparos.

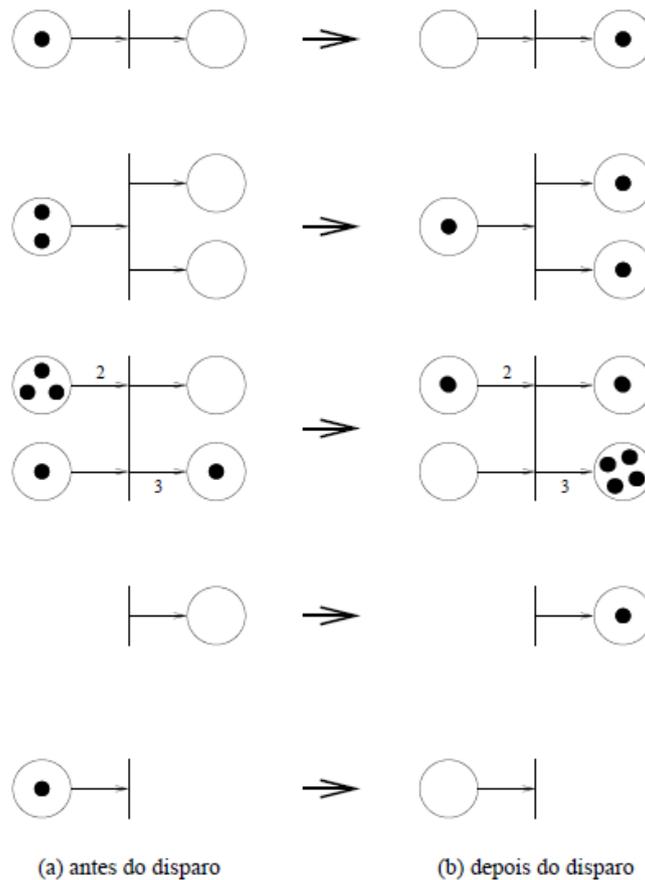



Figura 17. Efeitos dos disparos de transições

O desaparecimento das fichas nos lugares de entrada de *t* indica que as condições ou predicados associados àqueles lugares não são mais verdadeiros, e o surgimento de

fichas nos lugares de saída indica que os predicados associados a estes lugares são verdadeiros. O comportamento dinâmico do sistema é, assim, traduzido pelo comportamento da rede de Petri.

5.2.1 Definições Formais

Após a definição informal de uma rede de Petri, apresenta-se nesta seção a rede de Petri enquanto um modelo formal, que pode ser representada através de três visões [MUR89]:

- um grafo com dois tipos de nós e um comportamento dinâmico;
- um conjunto de matrizes de inteiros positivos ou nulos cujo comportamento dinâmico é descrito por um sistema linear;
- um sistema de regras baseado numa representação do conhecimento sob a forma condição → ação.

As visões gráfica e matricial diferenciam-se apenas pela forma de representação. Ambas permitem verificar se as transições são paralelas ou em conflito, se uma transição está ou não sensibilizada, bem como disparar uma transição e fazer evoluir a rede. Já a representação sob um sistema de regras tem o objetivo de compatibilizar a representação da rede de Petri com técnicas de Inteligência Artificial.

Embora a representação gráfica seja uma vantagem da rede de Petri, a característica mais importante deste modelo é o fato de ser formal. Sendo formal, é possível obter informações sobre o comportamento do sistema modelado, através da análise de suas propriedades, gerais ou estruturais. Desta forma, formalmente, uma rede de Petri pode ser definida como uma quíntupla (P, T, A, w, M_0), onde:

- $P = \{p_1, p_2, ..., p_m\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, ..., t_n\}$ é um conjunto finito de transições;
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos;
- $w : F \longrightarrow \{1, 2, ...\}$ é uma função que dá valor aos arcos;
- $M_0 : P \longrightarrow \{0, 1, 2, ...\}$ é a marcação inicial da rede, com $(P \cap T) = \emptyset$ e $(P \cup T) \neq \emptyset$.

Em um modelo de RdP, os estados estão associados aos lugares e suas marcações, e os eventos às transições. O comportamento de um sistema modelado por RdP é descrito em termos de seus estados e suas mudanças.

5.2.2 Vantagens da Aplicação de Redes de Petri na Gestão de Projetos

As redes de Petri oferecem uma série de vantagens na gestão de projetos. Estas estão descritas abaixo:

(1) Uma rede de Petri é capaz de modelar um sistema onde muitas atividades acontecem de forma simultânea e assíncrona. Ela permite modelar atividades concorrentes e seus conflitos. Além disso, pode auxiliar na identificação de bloqueios (*deadlocks*).

(2) Ela fornece informações ao gerente de projetos para ajudar a verificar e raciocinar sobre o progresso tardio das atividades.

(3) É capaz de regenerar e reprogramar as atividades do projeto. Uma rede de Petri é também uma representação dinâmica de um sistema e, portanto, é adequada para o monitoramento em tempo real.

(4) A lógica expressa em palavras pode ser transferida para uma forma gráfica e uma forma matemática, que são adequadas para análise do projeto.

(5) As redes de Petri fornecem resultados analíticos em conjunto com a flexibilidade da modelagem advinda da simulação.

(6) A simulação dinâmica de um projeto pode ser visualizada graficamente. Usando subredes, as partes do projeto podem ser facilmente modeladas. Assim, é possível simular todo o projeto e manter todas as subredes em primeiro plano e o restante em segundo plano.

(7) A rede de Petri é capaz de representar a interdependência de recursos, alocação parcial, substituição de recursos e exclusividade mútua.

(8) O planejamento do projeto pode ser melhorado usando algumas propriedades comportamentais das redes de Petri, tais como o alcance e a limitação.

(9) A rede de Petri pode ser simplificada pela combinação de lugares e transições semelhantes. Assim, o tamanho da rede pode ser reduzido usando redes de Petri de alto nível, tais como as redes de Petri coloridas (CPNs) ou as redes de Petri orientadas a objetos (OPNs).

(10) A modelagem de atividades fantasmas nos diagramas de rede do CPM/PERT costuma sercomplicada. No entanto, a representação das relações de precedência usando as redes de Petri torna esta atividade mais fácil de ser realizada.

5.2.3 Extensões das Redes de Petri

As tentativas de utilização das redes de Petri em problemas práticos reais mostraram duas desvantagens principais. Em primeiro lugar, não existe o conceito de dados e os modelos tornam-se excessivamente grandes porque toda a manipulação de dados tem que ser representada através da estrutura da rede (isto é, através de lugares e transições). Em segundo lugar, não existe noção de hierarquia e, portanto, não é possível construir um modelo de grande porte através de um conjunto separado de submodelos com interfaces bem definidas.

Para resolver estes problemas, foram propostas as redes de Petri de alto nível, onde se procura incorporar estruturas de dados e decomposição hierárquica ao modelo original das redes de Petri. Entre as redes de Petri de alto nível encontram-se as redes de Petri Coloridas Temporizadas (TCPNs).

5.2.3.1 Redes de Petri Coloridas Temporizadas (TCPNs)

Em ambiente de múltiplos projetos, o uso redes de Petri básicas poderá gerar um gráfico muito complexo para o problema da programação de múltiplos projetos com recursos limitados. Portanto, faz-se necessário utilizar redes de Petri de alto nível para combinar lugares e transições semelhantes, que são convertidos a partir do diagrama de redes para as redes de Petri em diferentes projetos concorrentes. Assim, o tamanho do gráfico gerado pode ser reduzido, tornando-o mais fácil de ser analisado. Entre as várias extensões para as redes de Petri básicas, a rede de Petri colorida temporizada [MUR89] é uma das redes de Petri de alto nível que permitem modelar sistemas dinâmicos a eventos discretos muito mais compactos e concisos.

Esta vantagem é obtida através da introdução de cores para distinguir as fichas que representam diferentes entidades e usando lógica adicional para controlar os fluxos de destas fichas. Assim, de acordo com estas vantagens, a rede de Petri colorida temporizada foi escolhida nesta pesquisa para modelar o problema da programação de múltiplos projetos com recursos limitados.

As definições sobre a redes de Petri coloridas temporizadas foram baseadas nas definições semânticas elaboradas por Jensen [JEN01]. A temporização das redes de Petri coloridas (CPNs) pode ser alcançada ligando o tempo a um local, transição, ou arco. Quando uma ficha é colocada em um lugar, pela ocorrência de um passo (*step*), esta se

mantém indisponível pelo período equivalente ao tempo de atraso desta ficha. Após este período, a ficha se torna disponível.

A rede de Petri Colorida Temporizada pode ser formalmente definida como:

$$TCPN = (\Sigma, P, T, A, C, G, E, M_0, t_0), \quad (1)$$

onde P = { $p_1, p_2, \cdots, p_n$ } é o conjunto finito de locais , n> 0 é o número total de locais, T = { $t_1, t_2, \cdots, t_m$ } é o conjunto finito de transições , m > 0 é o número total de transições , $P \cap T = \emptyset$, a é o conjunto finito de arcos , e os elementos de A são identificados pela expressão "$p_i - t_j$" ou "$t_j - p_i$" , $i \in n$, $j \in m$; Σ é o conjunto finito de cores, $\Sigma (pi) = \{ a_i, 1, a_i, 2, \cdots, a_i, u_i \}$, $\Sigma (t_j) = \{ b_j, 1, b_j, 2, \cdots, b_j, v_j \}$, $u_i = | \Sigma (p_i) |$ denota o número de cores sobre p_i ($i \in n$) = $v_j | \Sigma (t_j) |$ denota o número de cores sobre t_j ($j \in m$), $C: P \rightarrow \Sigma$ é um conjunto de funções de cores que mapeiam locais para o conjunto de cores, eles especificam atributos dasfichas em um lugar específico; G é um conjunto de funções de guarda definido a partir de T em funções booleanas , isto é, suas avaliações são verdadeiro ou falso, e evitam as fichas de cores específicas de fluir através de uma transição, E é um conjunto de funções de de entrada e saída dos arcos; M_0 é um conjunto de marcações iniciais de lugares, t_0 é o tempo de início do modelo.

A marcação de um lugar (*p*) denotado por *M(p)*, é composto por um conjunto de fichas com suas informações individuais de tempo. Ela representa os estados realizados entre os estados possíveis, indicados pelas cores do lugar *p*. Esta pode ser alterada pelo disparo de uma transição habilitada no qual o lugar *p* é um local de entrada ou de saída.

5.3 Mecanismo de Mapeamento entre o Diagrama de Rede e a Rede de Petri

Kim et al [KIM95] estudaram uma espécie de mecanismo de mapeamento entre ométodo do caminho crítico (CPM) e as redes de Petri com base em redes básicas de Petri no ambiente de um único projeto. Reddy et al [RED01] propuseram outro mecanismo de mapeamento entre PERT e redes básicas de Petri em situação de um único projeto. O pontoem comum destas duas pesquisas é que as atividades do diagrama de rede foram mapeadas em transições temporizadas e eles discutiram o mecanismo de mapeamento somente em ambientes de um único projeto.

No entanto, existe uma lacuna crítica nestes dois mecanismos. Após o disparo de umatransição, as fichas são removidasimediatamente a partir do local de entrada desta transição. Entretanto,as fichas não podem ser depositadas no local de saídacorrespondente porque estas devem se manter indisponíveis até que o intervalo de

tempo associado com a transição encerre. Portanto, neste intervalo de tempo, o estado do sistema de marcações é desconhecido e um mecanismo de rastreio adicional deve ser estabelecido para controlar o movimento das marcações, o que o torna difícil de analisar o comportamento dinâmico do sistema. Ainda, há necessidade de criar um lugar fictício de destino para a última transição temporizada (que representa o tempo de execução da última atividade do projeto).

Esta pesquisa propõe um novo mecanismo de mapeamento para resolver este problema. Neste mecanismo, uma atividade num projeto é convertida em um lugar cronometrado, e um evento é convertido em uma transição instantânea, de modo que o mecanismo de rastreio das fichas pode ser omitido. Caso contrário, um diagrama de rede de um projeto apenas mapeia uma rede de Petri no ambiente de um único projeto. Entretanto, para o ambiente multiprojetos, que consiste em diversos projetos executados ao mesmo tempo, a conversão para redes de Petri será extremamente complicada e resultará em um gráficocomplexo. Portanto, o conceito de cores é adotado nestapesquisa para converter os diagramas de rede de múltiplos projetos em uma espécie de TCPN onde projetos diferentes serão representados por cores diferentes, de modo que a complexidade eo tempo de modelagem podem ser reduzidos.

A Figura 18 mostra o contraste entre o diagrama de rede e a TCPN.

(a) Diagrama de redes (b) Diagrama de redes de Petri coloridas temporizadas

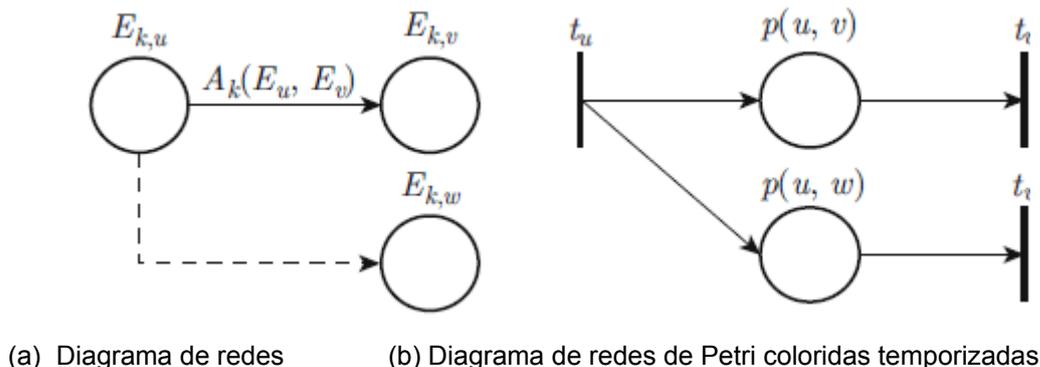



Figura 18. Conversão do diagrama de redes para a TCPN

Assim, a TCPN é gerada conforme os seguintes passos:

(1) Um evento $E_{e,i}$será mapeado para uma transição instantânea t_i (que é uma transição sem duração de tempo, onde $i = 1, 2, \cdots, M´$).

(2) Uma atividade A_k (E_u, E_v) será mapeada para um lugar colorido p (E_u, E_v), cujas cores também podem representar atividades, restrição de recursos, e etc.

(3) A restrição de tempo de uma atividade será mapeada para o arco de entrada do lugar *p*.

(4) O mecanismo de mapeamento da atividade fictícia A'_k (E_u, E_w) é o mesmo que a das outras atividades, mas não há nenhuma restrição de tempo no arco de entrada.

(5) A direção da seta da TCPN é consistente com a do diagrama de rede.

Para efeito deste conjunto de passos para a geração da RdP, observou-se a necessidade das seguintes deduções:

- Dedução 1: A hora de início mais cedo (EST) da atividade A_k (E_u, E_v) é o momento em que uma ficha é depositada para o lugar p (E_u, E_v) da TCPN.
- Dedução 2: O tempo de término mais tarde (LFT) da atividade A_k (E_u, E_v) é o momento em que umaficha é removida do lugar p (E_u, E_v).

Desta forma, a rede de Petri que corresponde às informações contidas na WBS (tais como, atividades, restrições de tempo, alocações de recursos, etc.) é gerada através das seguintes regras:

1. Cada atividade do projeto é transcrita como um lugar, umatransição e um arco de entrada ligando o lugar à transição (esta ocorrência é ilustrada pelaFigura 19). O lugar que representa a atividade inicial do projeto é o lugar i.

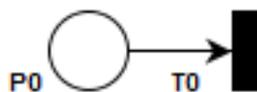



Figura 19. Representação de uma atividade do projeto para uma RdP

2. O ponto de início do projeto é definido pela alocação de uma ficha em um lugar da RdP. Esta ficha é depositada no início do fluxo, ou seja, no lugar i.
3. Um ponto no projeto onde diferentes fluxos de atividades paralelas convergem em um único fluxo de atividades (para que a execução do fluxo prossiga, é necessário que todos os fluxos paralelos que convergem para a sincronização tenham sido completados), é representado por uma transição e n lugares de entrada. A Figura 20 apresenta esta ocorrência.

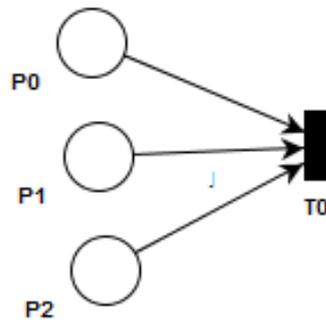



Figura 20. Representação de convergência de lugaresem uma RdP

4. Um ponto no projeto onde uma atividade é predecessora de duas ou mais atividades é representado através de uma transição imediata (relembrando que o tempo gasto para a realização da atividade está registrado no lugar que corresponde a esta atividade). Assim, ligadas ao lugar de saída há n transições imediatas, responsáveis pela representação de cada fluxo a ser seguido, a Figura 21 apresenta esta ocorrência.

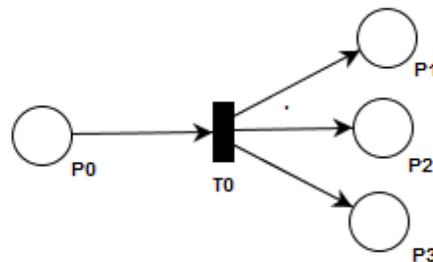



Figura 21. Representação de fluxos paralelos em uma RdP

5. Cada agrupamento de atividades, tal como uma fase, não são considerados na geração da rede de Petri.

6. O recurso alocadoparauma determinada atividade é representado por uma ficha em um lugar de recurso (o qual deve ser ligado até a transição que representa a atividade por um arco restaurador). Assim, o rótulo deste lugar corresponde ao nome do recurso alocado para uma atividade do projeto (por exemplo, Ana). A Figura 22 apresenta esta representação.

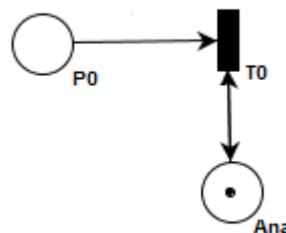



Figura 22. Representação de um recurso executor de uma atividade para uma RdP

Pode-se verificar que cada construção é iniciada com lugar(es) e terminada com transição(ões), possibilitando que seja simplificado o arranjo das construções. Desta forma, os arcos correspondem diretamente às relações de precedência presentes em um projeto de software. O tempo em que cada atividade é executada está relacionado ao tempo de espera em cada estado da rede. Assim, quando uma atividade é concluída, a próxima atividade deve ser executada após o tempo de espera determinado pela rede.

5.4 Petri Net Markup Language

Ausência de um padrão para representação das redes de Petri e a popularização do uso de ferramentas de modelagem resultou no desenvolvimento de diferentes formatos de representação destas redes. A colaboração entre estas diferentes ferramentas poderia trazer diversos benefícios [BIL03], ampliando as possibilidades de simulação e validação dos modelos desenvolvidos. Com base neste cenário, a comunidade acadêmica tem realizado esforços na criação de um formato de representação padrão intercambiável entre ferramentas de modelagem com redes de Petri. Uma das propostas que setornou bastante popular é a da linguagem PNML (*Petri Net Markup Language*) [WEB02].Trata-se de um padrão baseado em XML (*eXtensible Markup Language*) [WCG04] que tem comoobjetivo representar as características comuns a todos os tipos de redes de Petri. Embora a PNML ainda não seja aceita por todas as ferramentas existentes, observa-se um tendência para a adoção deste formato. Este formato foi utilizado como o padrão pelo protótipo de ferramenta implementado nesta tese.

Esta seção apresenta as características básicas da linguagem PNML, assim como as extensões que foram adicionadas para representar as redes temporizadas e coloridas.

5.4.1 Características Básicas da PNML

Três princípios foram levados em consideração na definição do padrão PNML pelos seus autores [WEB02]:

- Flexibilidade: o formato deve ser capaz de representar qualquer tipo de rede de Petri, com suas extensões e características especificas. Na PNML, as informações adicionais podem ser anexadas à rede propriamente dita, ou apenas aos elementos relacionados àquela informação.
- Não-Ambiguidade: significa afirmar que a RdP representada e seu respectivo tipo podem ser determinados unicamente pela sua representação em PNML.

- Compatibilidade: garante que o máximo de informação possível seja compartilhado entre os diferentes tipos de rede de Petri existentes.
- Legibilidade: o formato deve permitir a leitura e edição das redes em qualquer editor de textos.

A PNML permite que, através de programas tradutores, a rede possa ser transferida entre as diversas ferramentas (vide Figura 23). A PNML aceita, desta forma, a comunicação entre diferentes formatos de representação de redes de Petri. Logo, a PNML disponibiliza mecanismos de validação de documentos nesse formato utilizando DTDs (*Document Type Definitions*) associados a verificadores sintáticos de documentos XML. Na PNML, estas DTDs são chamadas de PNTD (*Petri Net Type Definition*) e são definidos através de *XML Schemas*na linguagem *RELAX NG* [OAS01].

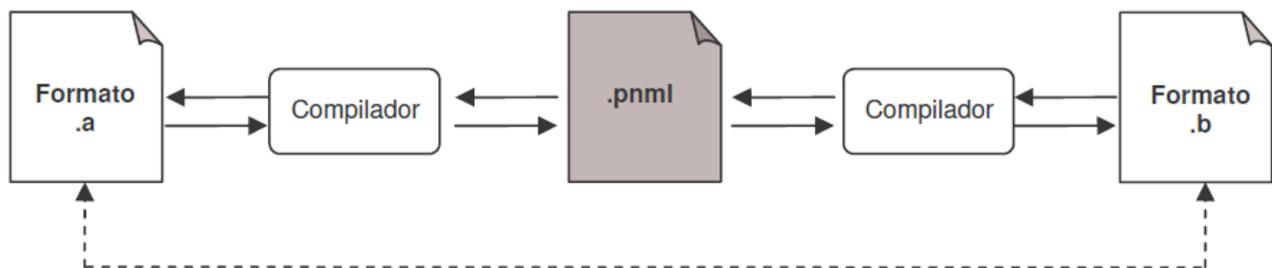



Figura 23. Comunicação entre diferentes formatos de representação para redes de Petri – adaptado de [WEB02]

5.4.2 Estrutura de um Documento PNML

Um único arquivo PNML pode conter a descrição de várias redes de Petri. Cada rede de Petri possui um identificador único dentro do arquivo e um tipo como atributos. O tipo fornecido comoatributo deve ser um URI (*Uniform Resource Identifier*) para o arquivo de definição do tipocorrespondente. A Figura 24exemplifica a estrutura de um PNML com duas redes, chamadas de “net1” e“net2”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<pnml>
  <net id=”net1”
     type=”http://www2.informatik.hu-berlin.de/ptNetb.pntd”>
  </net>
  <net id=”net2”
     type=” http://www2.informatik.hu-berlin.de/ptNetb.pntd”>
  </net>
</pnml>
```

Figura 24. Duas redes representadas em PNML [WEB02]

As tags `<pnml>` e `</pnml>`delimitam o conteúdo do arquivo PNML. Cada rede dentro do arquivo é delimitada pelas tags `<net>`e `</net>`.Uma rede contém vários objetos que representam a sua estrutura: seus lugares, transições, arcos e quaisquer outros objetos relacionados a algum tipo específico de rede de Petri. Cada objeto possui um identificador único que o referencia dentro da rede.

Um lugar é definido em PNML pela tag `<place>`. A Figura 25 demonstra uma declaração de um lugar na rede. Este lugar tem “p1” como seu identificador, está disposto na posição (x=10, y=10) do ambiente gráfico, possui um rótulo “Place1” colocado na posição (x=-5, y=-2) e possui um *token* como marcação inicial.

As informações gráficas de qualquer objeto em PNML são fornecidas pelo elemento `graphics`. Seu conteúdo varia de acordo com o objeto em que está contido. Em geral podem ser definidas as informações sobre a posição, o tamanho e as cores. Na Figura 25 é definida a posição do lugar através da tag `<position>`. A marcação inicial do lugar é definida com a utilização da tag `<initialMarking>`.

```
<place id="p1">
  <graphics>
     <position x="10" y="10" />
  </graphics>
  <name>
     <text>Place1</text>
     <graphics>
       <offset x="-5" y="-2" />
     </graphics>
  </name>
  <initialMarking>
     <text>1</text>
  </initialMarking>
</place>
```

Figura 25. Exemplo de descrição de lugar com PNML

As transições são identificadas pela tag `<transition>`. A Figura 26demonstra a definição deuma transição acompanhada de um rótulo “t”.

```
<transition id="t1">
  <graphics>
      <position x="30" y="10" />
      <dimension x="8" y="2" />
  </graphics>
  <name>
      <text>t</text>
      <graphics>
        <offset x="-5" y="-2" />
      </graphics>
  </name>
</transition>
```

Figura 26. Exemplo de descrição de transição com PNML

Os arcos são definidos através de um identificador e dos seus elementos de origem e destino,utilizando-se a tag `<arc>`. AFigura 27mostra um exemplo de arco.Pode-se definir um conjunto de pontos pelos quais o desenho do arco deve passar (utiliza-se o elemento `graphics` para definir estas informações).

Outro atributo importante de um arco é a sua inscrição, que pode definir o seu peso (comoem redes *Place/Transition*) ou sua expressão (como nas redes Coloridas). Para oferecer maiorflexibilidade, a inscrição é definida como um elemento de texto, utilizando-se por isso a tag `text`.

```
<arc id="a1" source="p1" target="t1">
  <graphics>
      <position x="20" y="10" />
      <position x="25" y="10" />
  </graphics>
  <inscription>
      <text>2</text>
      <graphics>
        <offset x="5" y="-2" />
      </graphics>
  </inscription>
</arc>
```

Figura 27. Exemplo de descrição de arco com PNML

Utiliza-se a tag `<toolspecific>` para representar informações especiais, que são dependentes de uma determinada ferramenta.Assim, o conteúdo de um elemento `toolspecific` é definido pela ferramenta correspondente, conformesuas necessidades. Um exemplo de conteúdo específico é apresentado na Figura 28.

```
<transition id="t1">
  <name>
     <text>checkup</text>
     <graphics>
        <offset x="-5" y="-2" />
     </graphics>
  </name>
  <toolspecific tool="example" version="1.0">
      <action type="human" cost="3" />
  </toolspecific>
</transition>
```

Figura 28. Exemplo de utilização da *tagtoolspecific*

Conforme ilustrado acima, uma ferramenta hipotética chamada “example” associa a umatransição informações sobre a ação que ela representa. Neste caso, a transição representa umaação executada por um agente humano, e que tem um certo custo associado igual a 3. Observeque esta informação é associada à rede de Petri, mas não é de interesse de outras ferramentas.Apenas a ferramenta “example” pode compreender o seu significado.

5.4.3 Exemplo de uma Rede de Petri representada através da PNML

AFigura 29 representa graficamente uma rede de Petri onde a transição “T1” tem um lugar de entrada (P1) e um lugar de saída (P2). Quando “T1” dispara, ela remove (consome) um *token* de todos os seus lugares de entrada e adiciona (produz) um outro *token* em todos os seus locais de saída.

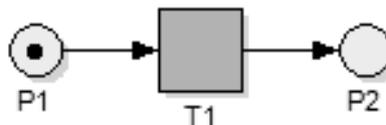



Figura 29. Rede de Petri com dois lugares e uma transição

Na Figura 30, a transição “T1” consumiu um *token* do lugar “P1” e produziu outro no lugar “P2”. Comportamentos mais complexos podem ser modelados pela adição de mais locais e transições no diagrama. Transições também podem ter tempos associados (redes de Petri temporizadas) e os *tokens* podem ter diferentes valores e atributos ao longo da rede (redes de Petri coloridas). Entretanto, é possível simular processos utilizando apenas os elementos básicos de redes de Petri.

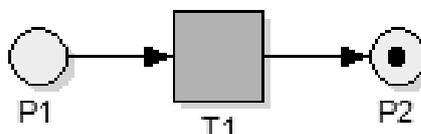



Figura 30. Resultado após disparo da transição T1

A Figura 31 contém a descrição em PNML da rede apresentada na Figura 29. As informaçõesgráficas não estão presentes para simplificar o exemplo.

```
<pnml xmlns="http://www.example.org/pnml">
  <net id="n1">
    <place id="p1"/>
    <arc id="a1" source="p1" target="t1"/>
    <transition id="t1"/>
    <arc id="a2" source="t1" target="p2"/>
    <place id="p2"/>
  </net>
</pnml>
```

Figura 31. Um exemplo simplificado do formato PNML

5.5 Considerações Finaisdo Capítulo

Este Capítulo apresentou o estudo realizado nesta tese de doutorado sobre a utilização de redes de Petri na gestão de projetos. Inicialmente, foram apresentados o vocabulário e conceitos utilizados na modelagem de sistemas a eventos discretos. Em seguida, foram discutidas principais vantagens identificadas na aplicação das redes de Petri na gestão de projetos de software. Após, foi apresentado o mecanismo de mapeamento entre o diagrama de rede e a rede de Petri utilizado nesta tese. Finalmente, foram apresentados os conceitos relacionados à linguagem PNML (Petri Net Markup Language).

O próximo Capítulo discorre sobre o modelo de reconfiguração dinâmica de projetos de software desenvolvido nesta tese, que considera o uso de redes de Petri para simular diferentes cenários que ocorrem em decorrência de eventos inesperados.

6 MODELO PARA A RECONFIGURAÇÃO DINÂMICA DE PROJETOS DE SOFTWARE

6.1 Modelo de Referência para Reconfiguração Dinâmica de Projetos de Software

Considerando a associação da proposta desta pesquisa com o Modelo de Reconfiguração Dinâmica para Projetos de Desenvolvimento de Software [CAL10], seu funcionamento é detalhado a seguir.

O processo de reconfiguração no modelo proposto por Callegari [CAL10], inicia por meio da ocorrência de eventos (que são disparados pela criação, alteração ou remoção de um elemento de projeto pelo gerente). Como exemplos destes eventos citam-se: a criação de uma nova tarefa no projeto, o aumento ou decréscimo do prazo em uma tarefa, e saída de um recurso do projeto. Assim, quando um evento de reconfiguração ocorre, várias tarefas podem ser afetadas. Dessa forma, o modelo segue um ciclo para chamadas de propostas (*Call For Proposals* - CFP) onde, inicialmente, verificam-se as posições de trabalho para as tarefas que foram afetadas em função da ocorrência do evento. Através dessa CFP, é possível identificar os recursos capazes de ocupar cada posição de trabalho.

Uma vez composta a CFP, as mesmas são repassadas aos componentes do modelo denominados *solvers*, os quais têm como função gerar propostas para a realocação de recursos. Nesse sentido, os *solvers* recebem a CFP, a analisam e assim propõem alternativas de solução que possibilitem dar continuidade a execução do projeto.

O modelo permite a execução de vários *solvers* em paralelo, que podem competir entre si durante o processo de submissão de propostas. Desta forma, o modelo original limita-se a assumir como comportamento padrão o envio de uma proposta pertencente a um recurso para uma posição de trabalho que esteja aberta. A Figura 32 apresenta uma visão geral do funcionamento do modelo de referência proposto em Callegari [CAL10].

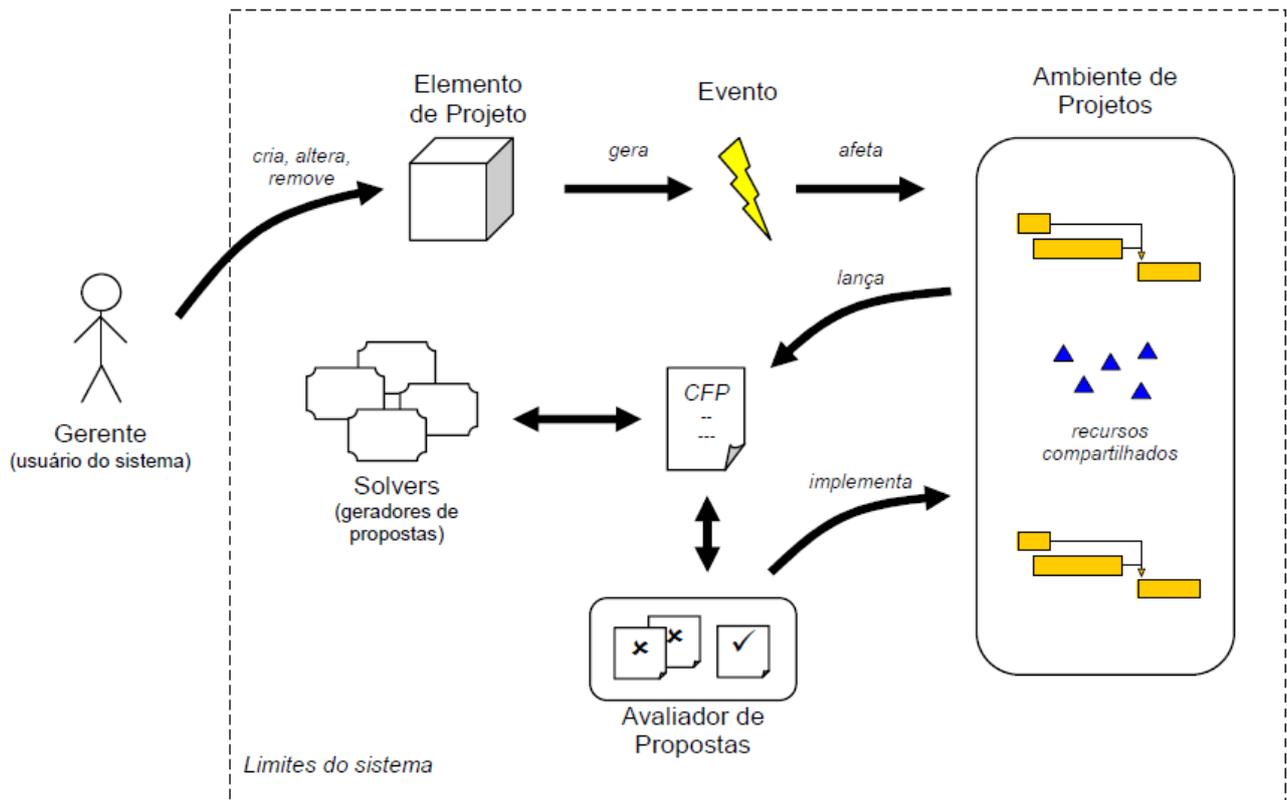



Figura 32. Funcionamento geral do modelo de reconfiguração [CAL10]

Após a etapa de geração de propostas, o modelo segue seu fluxo e parte para a avaliação das mesmas. A partir disso, algumas propostas são implementadas e outras descartadas. Com base nessa implementação alguns recursos podem ser realocados, ou tarefas podem ser reprogramadas. Callegari [CAL10] salienta que, para algumas situações a implementação de uma proposta pode incidir na geração de novos eventos no modelo, fato que ocasiona o início de um novo ciclo de execução para buscar novas alternativas de solução.

Outro conceito importante que envolve o modelo está relacionado à proposta do recurso para ser alocado às tarefas componentes de um projeto. Esta proposta é calculada através de uma fórmula, incluída no modelo de reconfiguração [CAL10], cujo objetivo é estabelecer um parâmetro numérico que traduza as decisões na realocação dos recursos. A assinatura de função para a fórmula presente no modelo é dada pela expressão em (2) [CAL10].

Valor Proposto = Função (recurso, posição de trabalho, duração proposta) (2)

A assinatura informa que a proposta é calculada em função do recurso proponente, da posição de trabalho e da duração estimada para execução da tarefa, sob a ótica de uma representação de custos, onde quanto menor for o custo calculado melhor é a

proposta. O procedimento de cálculo das propostas é fazer que as ofertas dos recursos decresçam o valor de uma tarefa de forma proporcional às características individuais de cada recurso. A fim de exemplificar tal proposição considere que a unidade de tempo (u.t.) que compõe a tarefa receba um valor padrão de custo no valor de 100,00. Logo, uma tarefa com a duração de quatro u.t. e duas posições de trabalho teria um custo total para o projeto dada pela expressão em (3) [CAL10].

$$4 \times 2 \times 100{,}00 = 800{,}00 \qquad (3)$$

Callegari [CAL10], desta forma, sugere que a medida que os recursos preencherem essas posições, o custo total da tarefa deva ser decrementado. Assim, caso os recursos candidatos a ocuparem as posições corresponderem exatamente ao perfil pretendido, o custo da tarefa se aproximará de zero.

Uma vez esclarecidos os elementos presentes na fórmula sugerida por Callegari [CAL10] e a idéia para o procedimento de cálculo, apresenta-se na Figura 33 a composição da fórmula padrão deste modelo de referência para reconfiguração dinâmica de projetos de desenvolvimento de software.

$$\boldsymbol{ValorFinal = ValorInicial - Descontos + Penalidade}, \text{ sendo que:}$$

$$ValorInicial = CustoPadrãoDeUmaUnidadeDeTempo \times DuraçãoDaTarefa$$

$$Descontos = DuraçãoDaTarefa \times \sum_{n}^{i=1} (F_i \times E_i)$$

n = número de elementos E (neste caso, $n=4$)

F_i = fator de importância relativa para o elemento i

E_i = valor normalizado calculado para o elemento i

$Penalidade = F_p \times StartupPenalty(recurso, posição)$

F_p = fator de importância para a penalidade.

Figura 33. Composição da fórmula padrão do modelo de referência para reconfiguração dinâmica de projetos de desenvolvimento de software [CAL10]

Os quatro elementos normalizados nesta fórmula (E_1 a E_4) são:

- E1 = o grau de adequabilidade do recurso à posição de trabalho;

- E2 = o Índice Objetivo de Alocação dos Recursos (IOBJALOC)[2];
- E3 = a prioridade do projeto ao qual a tarefa pertence;
- E4 = a criticidade da tarefa (se é crítica ou não);

Assim, esta fórmula é compartilhada por todos os recursos e proposta por eles, para que o modelo possa decidir a realocação de recursos.

A proposta elaborada por Schlösser [SCH10], entretanto, apresenta um *solver* baseado na abordagem de sistemas multiagentes. Nesta solução, os agentes gerenciam suas agendas e interagem entre si a fim de possibilitar o envio de propostas de alocação quando ocorre o recebimento de CFP's. O modelo de modelo de referência para reconfiguração dinâmica de projetos de desenvolvimento de software [CAL10] permite a adição de novos *solvers*, de modo que o *solver* proposto por Schlösser [SCH10] é detalhado a seguir.

A arquitetura da solução proposta por Schlösser [SCH10] compreende duas classes internas de agentes: Agente Gerenciador de Propostas e Agente Recurso. Na classe Agente Gerenciador de Propostas existe apenas uma instância desse agente. Já na classe Agente Recurso, existe o número de instâncias correspondentes aos recursos disponíveis. O agente Gerenciador de Propostas tem como atribuição principal receber uma CFP e encaminhar as posições de trabalho contidas nela aos agentes componentes da classe Agente Recurso e aguardar o recebimento de propostas.

A execução deste processo, ilustrada na Figura 34, pode ser assim resumida: ao receber uma CFP o Agente Gerenciador de Propostas realiza um ordenamento sobre as tarefas nela contidas, para que se tenha um controle do recebimento de propostas geradas pelos Agentes Recurso. Em seguida, durante a fase Geração de Propostas, inicia-se um ciclo baseado no Protocolo de Rede de Contratos [SMI80], onde o Agente Gerenciador de Propostas encaminha os convites para propostas de agendamento de tarefas aos Agentes Recurso. Os Agentes Recurso encaminham respostas a esses convites ao Agente Gerenciador de Propostas. A cada iteração o Agente Gerenciador de Propostas verifica a viabilidade das propostas recebidas em função das dependências das tarefas e da sincronização de propostas. Após completar esse ciclo de execução, o Agente Gerenciador de Propostas analisa as respostas geradas pelos Agentes Recurso,

---

[2] IOBJALOC diz respeito ao índice de alocação de cadarecurso. Procura-se manter esse índice de alocação nafaixa de 90 a 100% de forma a não sobrealocar orecurso [CAL10].

ajustando-as nas prerrogativas do modelo de reconfiguração [CAL10], e assim as encaminha para que sejam analisadas neste modelo.

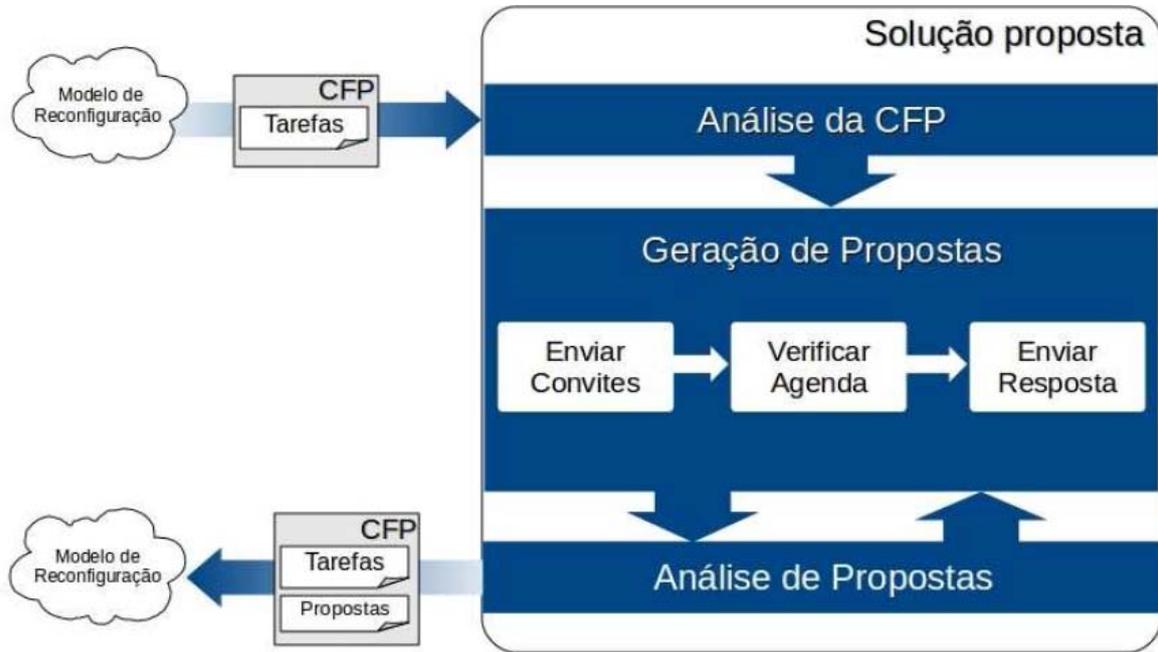



Figura 34. Processo para geração de propostas usando arquitetura multiagentes [SCH10]

Desta forma, após completar esse ciclo de execução, o Agente Gerenciador de Propostas analisa as respostas geradas pelos Agentes Recurso, ajustando-as nas prerrogativas do modelo de reconfiguração [CAL10], e assim as encaminha para que sejam analisadas neste modelo.

Após essa visão geral sobre o funcionamento do modelo de reconfiguração, apresenta-se a modelagem conceitual correspondente, proposto por Callegari [CAL10].No modelo, vide Figura 35, há um componente fundamental chamado `ProjectElement`, que representa um elemento de projeto. Os recursos, as atividades, os artefatos e até mesmo os próprios projetos são tipos especiais de elementos de projeto (a saber: `Resource`, `Activity`, `WorkProduct` e `Project`). As atividades possuem restrições de dependência entre si (definido através da classe `ActivityDependence`).

Este modelo, segundo o autor, está focado nos elementos considerados fundamentais para a especificação do modelo de reconfiguração. Desta forma, no modelo destacam-se os elementos que representam os papéis desempenhados pelos recursos (classe `Role`) e também as suas respectivas habilidades (classe `Skill`).

Todo elemento de projeto (`ProjectElement`) é definido como uma possível fonte (origem) de eventos. Tal comportamento é definido pela interface `EventSource`. Qualquer instância de `EventSource` pode, portanto, provocar um evento de

reconfiguração (`ReconfigurationEvent`). Para exemplificar: um projeto pode mudar de prioridade, um recurso pode se ausentar por um período, ou uma atividade pode ter suas estimativas revisadas. Ainda, os eventos de reconfiguração foram classificados como internos ou externos (`InternalEvent` e `ExternalEvent`).

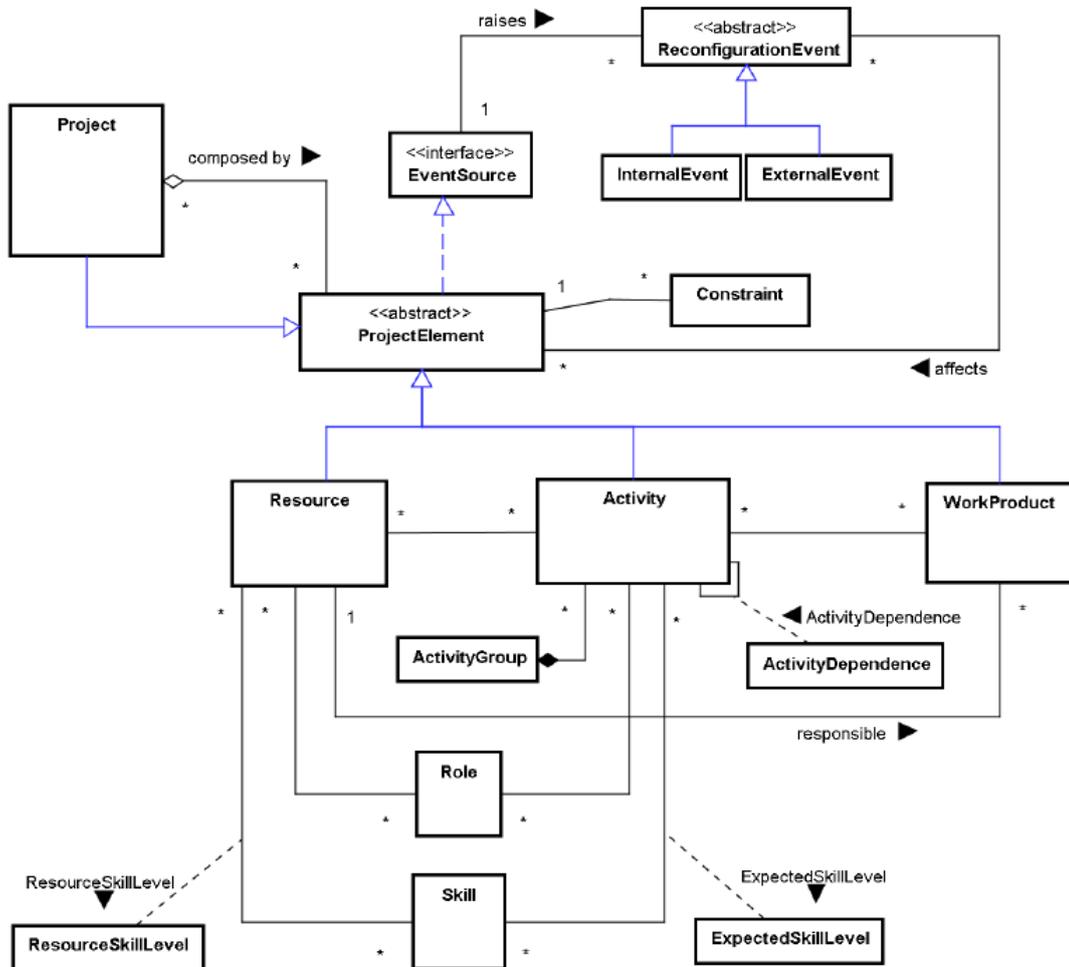



Figura 35. Modelo Conceitual de Eventos e elementos associados [CAL10]

Callegari [CAL10] justifica que o modelo conceitual apresentado acima é composto apenas pelos elementos considerados essenciais para a reconfiguração de projetos. Conforme o autor, a inclusão de todos os elementos apenas adicionaria complexidade à explicação, de forma que muitos elementos foram considerados secundários.

Observa-se, entretanto, que este modelo contempla de forma sucinta a integração das atividades dos projetos de software com os fluxos organizacionais. Outro ponto importante é que esta proposta específica não aborda profundamente os problemas relacionados ao sequenciamento das atividades de um projeto de software. Finalmente, observa-se que o modelo original não considera a simulação de cenários (com o auxílio de um mecanismo de simulação, tal como as redes de Petri).

6.2 Modelo Proposto

As empresas de software, geralmente, fazem uso dos conhecimentos relacionados à gestão de projeto para a construção de suas soluções, objetivando respeitar os critérios de qualidade acordados e as limitações de tempo, escopo e de recursos. O gerente geralmente cria um plano de projeto para especificar e limitar o escopo do projeto, descrevendo a estrutura analítica do projeto (WBS) e o cronograma do projeto. Para criar o cronograma do projeto, o gerente começa pelo conjunto de tarefas da WBS [PRE09]. Então, ele especifica todas as informações relacionadas ao projeto, tais como as tarefas individuais, a sequência em que as tarefas precisam ser realizadas, quantas tarefas podem ser executadas em paralelo, e os recursos para executar essas tarefas.

Durante o ciclo de vida de um projeto, dados atualizados, como o tempo ou os recursos que foram gastos para executar uma tarefa específica, são coletados e registrados pelo gerente do projeto. No entanto, o gerente pode não ter todas as informações relevantes na sua frente, forçando-o a interagir com outros departamentos na organização (como o departamento de recursos humanos).

Assim, o fluxo de atividades em um projeto individual geralmente está relacionado a outros fluxos comuns de atividades da organização (chamados de fluxos organizacionais). Ambos os tipos de fluxos de trabalho são executados em paralelo, têm seus próprios recursos e pode influenciar o tempo de atividades e custos do projeto de software (ver Figura 36).

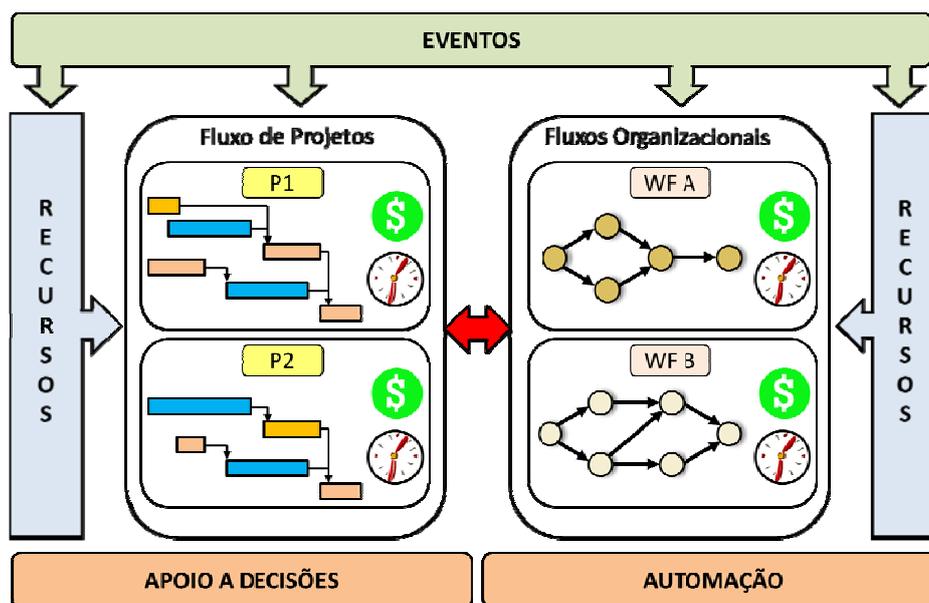



Figura 36. Interdependência entre os fluxos de trabalho organizacionais e o fluxo de projetos de software – adaptado de [ROS13]

Considerando esta situação apresentada acima, a fim de contribuir para a solução das dificuldades apontadas, uma primeira versão de um modelo computacional chamado *Software Planning Integrated Model* - SPIM foi proposta [ROS08a]. O SPIM foi inicialmente desenvolvido considerando a integração de conceitos de gerenciamento de projetos previstos no PMBOK com os conceitos de desenvolvimento de software advindos do *Rational Unified Process*- RUP [KRU00] e do *Object-oriented Process, Environment and Notation*- OPEN [FIR02]. Entretanto, foi possível identificar que os metamodelo de integração PMBOK+RUP e PMBOK+OPEN apresentavam uma estrutura de classes similar, substituindo apenas o pacote referente ao processo de desenvolvimento de software. Assim, tanto as classes dos pacotes `PMBOK` e `Common` como os relacionamentos entre as classes pertencentes a estes dois pacotes permaneceram inalterados. Desta forma, uma vez que o SPEM 2.0 é um metamodelo desenvolvido pelo OMG para a definição de processos de desenvolvimento de software e seus componentes, esta pesquisa avançou para a criação do modelo PMBOK+SPEM.

Faz-se necessário, desta forma, também definir os termos utilizados no presente estudo para os principais elementos de um processo de software. Isso porque, mediante a existência de vários processos de software na literatura, nem sempre o nome dos elementos são os mesmos, sendo comum o fato de termos diferentes terem o mesmo significado. Nesta pesquisa, assume-se a utilização dos termos apresentados no metamodelo SPEM, na sua versão 2.0 [OMG12], que são: Atividade, Artefato, Papel e Tarefa. Contudo, como outros processos de software são utilizados em alguns pontos deste trabalho, a Tabela 7 apresenta a correlação entre os termos utilizados nos processos RUP e OPEN com os termos do metamodelo SPEM.

Tabela 7: Principais terminologias utilizadas nos processo de software

SPEM 2.0	**Artefato**	**Papel**	**Atividade**	**Tarefa**
RUP	Artefato	Papel	Fase / Iteração / *Workflow Detail*	Atividade
OPEN	Produto de Trabalho	Produtor	Atividade / Técnica / *Workflow* / *Stage*	Tarefa

O modelo SPIM (ver Figura 38) foi concebido considerando a necessidade dos gerentes de projetos em acessar as informações advindas de outros departamentos da organização durante o planejamento de projetos de software. Para suportar esta funcionalidade, este modelo define três diferentes tipos de atividades:

(1) atividades produtivas: atividades diretamente relacionadas com a construção do produto de software;

(2) atividades gerenciais: atividades que são necessárias somente para coordenar a construção do produto de software; e

(3) atividades gerenciais de apoio: quaisquer outras atividades que não pertencem ao fluxo de atividades de um projeto individual (e pode ser mais partilhado por outros projetos).

A modelagem do banco de dados de um aplicativo de software é um exemplo de atividade produtiva. Organizar e conduzir uma reunião de acompanhamento do projeto é um exemplo de atividade gerencial. Esses dois primeiros tipos de atividades fazem parte do fluxo de trabalho do projeto. A contratação de um administrador de banco de dados (atividade normalmente realizada pelos departamentos de recursos humanos) é um exemplo de atividade gerencial de apoio. Após essa definição, foi possível distinguir quais atividades deveriam ser atualizadas por outros setores da organização (usando um mecanismo como uma ferramenta de *workflow*) e quais deveriam ser atualizadas diretamente pelo gestor do projeto.

O estudo da integração dos conceitos gerenciais advindos do PMBOK com o metamodelo do SPEM permitiu elaborar uma visão mais ampla de como a gerência de projetos pode positivamente contribuir no desenvolvimento de um produto de software. Assim, o metamodelo PMBOK+SPEM definido durante esta pesquisa fornece a estrutura conceitual necessária para o desenvolvimento de um modelo que auxilie o planejamento de projetos considerando os elementos que compõem o processo de desenvolvimento de software. Com este objetivo, foi desenvolvida a versão final domodelo computacional SPIM. A Figura 37 apresenta as etapas de desenvolvimento deste modelo.

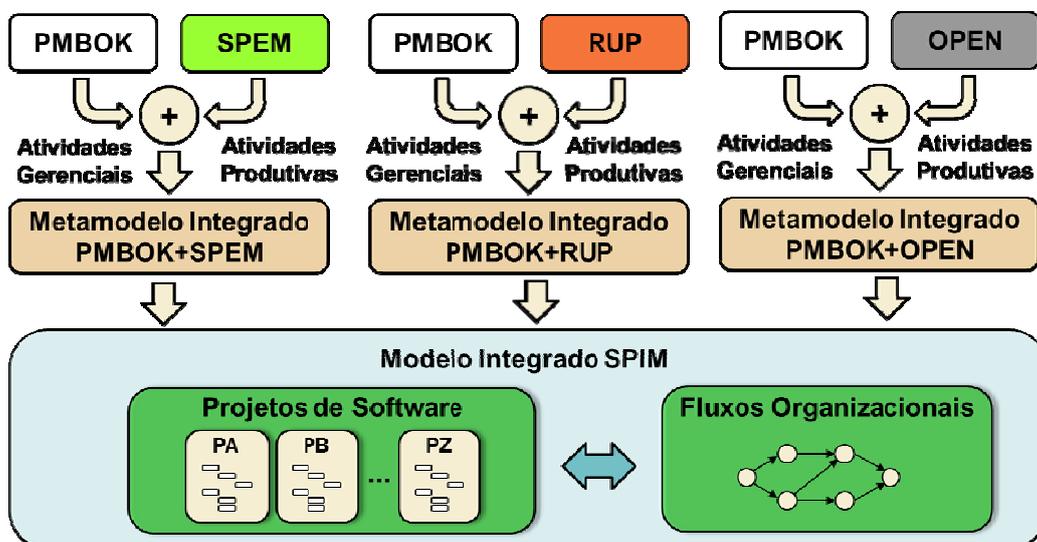



Figura 37. Etapas de desenvolvimento do SPIM

O modelo integrado SPIM, conforme ilustrado na Figura 38, considera a integração dos conceitos de gestão de projetos advindos do PMBOK com os conceitos de processos de desenvolvimento de software advindos do RUP, OPEN e SPEM. Ainda, este modelo considera a distinção entre as atividades dos projetos de software (atividades gerenciais e produtivas) das atividades dos fluxos organizacionais (atividades gerenciais de apoio).

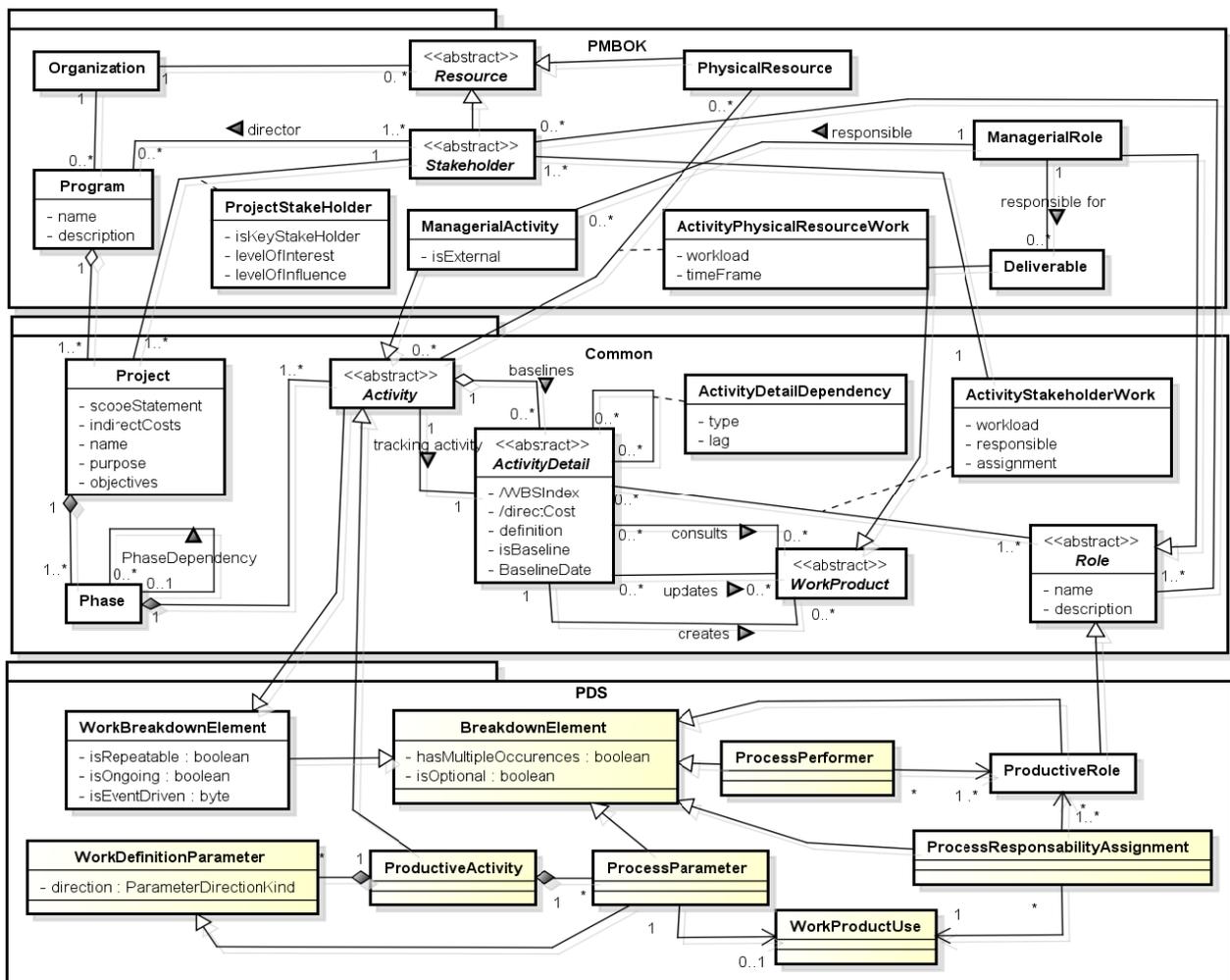



Figura 38. Classes do modelo integrado SPIM [ROS13]

Neste modelo, a classe `Organization` representa uma empresa que é organizada por programas (classe `Programs`). Os programas são grupos de projetos designados para alcançar um objetivo estratégico. As organizações, normalmente, dividem os projetos em várias fases (classe `Phase`) visando um melhor controle gerencial. Um recurso necessário para o projeto,tais como pessoas, equipamentos ou local, é representado pela classe `Resource`. Esses recursos são divididos em recursos ativos (classe `Stakeholder`) e não ativos (classe `PhysicalResource`). Além disso, a classe `PhysicalResource` representa um recurso físico num projeto, tal como um material

necessário para realizar uma atividade, um equipamento necessário para realizar uma atividade ou um local físico.

A classe `ProjectStakeholder` representa a relação do *stakeholder* com o projeto, por exemplo: se é um *stakeholder*chave do projeto, seu nível de interesse no projeto e seu nível de influência no projeto. A classe `ActivityPhysicalResourceWork` é associa zero ou mais recursos físicos para zero ou mais atividades (classe `Activity`). Ela estabelece a carga de trabalho físico recursos (atributo`workload`) nessa atividade. Os *stakeholders* podem desempenhar vários papéis (classe `Role`) durante a execução das atividades do projeto. Assim, para cada associação entre um papel e uma atividade (classe `ActivityStakeholderWork`), deve haver uma associação desta atividade com um *stakeholder* capaz de desempenhar esse papel. Assim, as atividades gerenciais são realizadas por papéis (`roles`) de gestão e atividades produtivas são realizadas por papéis produtivos.

Conforme mencionado anteriormente, o modelo proposto define três tipos diferentes de atividades. Atividades diretamente relacionadas com a construção do produto, tais como codificação ou modelagem de banco de dados, são chamadas de atividades produtivas (classe `ProductiveActivity`). Atividades gerenciais (classe `ManagerialActivity`), no entanto, podem pertencer ao fluxo de trabalho de desenvolvimento de software (atributo `isExternal = false`) ou pertencem ao fluxo organizacional (atributo `isExternal = true`). Cada atividade pode pertencer a uma ou mais *baselines*. Em cada geração de baseline, uma atividade deve estar relacionada a papéis e produtos de trabalho (classe `WorkProduct`). Assim, a classe `ActivityDetail` foi definida como responsável por manter estes relacionamentos, enquanto que a classe `Activity` foi definida como uma agregação de uma ou mais classes `ActivityDetail`. Além disso, a classe `ActivityDetailDependency` define se uma ou mais atividades podem ser executadas em paralelo, e, se duas atividades podem ser sobrepostas.

O *PMBOK Guide* representa as suas práticas em duas dimensões lógicas. Uma dimensão define nove áreas de conhecimento (classe `KnowledgeArea`), enquanto que a outra dimensão descreve trinta e nove processos de gestão de um projeto (classe `ManagementProcess`) que estão organizados em cinco grupos de processos (classe `ProcessGroup`). As áreas de conhecimento são classificadas em áreas centrais de conhecimento (classe`CoreKnowledgeArea`), tais como escopo, a equipe, custo e

qualidade, ou áreas de facilitação do conhecimento (classeFacilitatingKnowledgeArea), como recursos humanos, comunicações, riscos e dos contratos. A classe Artifact representa algo que é produzido, consumido ou modificado (como documentos, modelos ou códigos-fonte), durante a execução das atividades.

De acordo com o metamodelo SPEM, cujos conceitos encontram-se no pacote PDS, uma atividade suporta o agrupamento de lógico de elementos contidos na WBS (classe BreakdownElement). O auto-relacionamento definido para a classe Activity permite a reutilização do conteúdo definido para uma atividade em outra atividade. Assim, torna-se possível herdar uma estrutura definida para uma atividade (em termos de seus elementos) e aproveitá-los em uma segunda atividade. A relação entre a classe Activity e a classe ProcessParameter estabelece parâmetros de entrada e/ou saída para as atividades em termos de produtos de trabalho. A classe ProcessPerformer estabelece a relação entre as atividades e os papéis (classe ProdutiveRole) no projeto. A classe ProcessResponsabilityAssignment estabelece a relação de responsabilidade entre os papéis e produtos de trabalho.

A distinção proposta entre os tipos de atividades que ocorrem em um projeto de software permite aos gerentes de projeto identificar possíveis relações de dependência entre as atividades dos fluxos organizacionais e as atividades de um fluxo de trabalho de um projeto de software específico. Por exemplo, a atividade de desenvolvimento de um software para dispositivo móvel (que se encaixa no fluxo de trabalho do projeto) pode depender a aquisição de um dispositivo portátil pelo departamento responsável da empresa (esta atividade se encaixa nos fluxos organizacionais). A dificuldade em visualizar esta interdependência entre os fluxos de trabalho durante o planejamento das atividades pode afetar negativamente o projeto, resultando, por exemplo, o aumento dos custos e atrasos no cronograma do projeto. Como consequência, o gerente de projetos precisa de um suporte contínuo a fim de manter o controle desses tipos de dependências.

Para este estudo, cada fluxo organizacional é considerado como um conjunto de atividades que podem ser consumidos por um ou mais projetos de software. As atividades gerenciais de apoio fazem parte dos fluxos organizacionais. Assim, o SPIM permite que cada instância de um fluxo de trabalho organizacional seja registrada como uma atividade gerencial de apoio nos projetos de desenvolvimento de software. De acordo com Leymann e Roller [LEY00], a tecnologia de *workflow* é uma ferramenta poderosa que

pode ser beneficamente utilizada para o gerenciamento de projetos. Desta forma, um sistema de *workflow* pode constantemente atualizar e informar os projetos as informações de cada instância de fluxo organizacional.

A análise de como o conhecimento sobre gerência de projetos permite aperfeiçoar os processos de desenvolvimento de software atuais torna possível o desenvolvimento de novas ferramentas capazes de suportar diferentes níveis de automatização no planejamento e na execução de atividades num projeto de software. Por essa razão, o modelo integrado foi desenvolvido com os seguintes os objetivos:

- permitir o planejamento de projetos considerando os elementos que compõem a gerência de projetos e os processos de desenvolvimento de software;
- fazer a distinção dos tipos de atividades (gerencial ou produtiva) e produtos;
- adicionar a noção de disponibilidade a um recurso, de modo que permita automatizar os processos de alocação de recursos em projetos de software;
- prever a informação de carga de trabalho (*workload*) para as associações de um papel, atividade e *stakeholder*, preparando para a automatização; e
- distinguir as possíveis relações entre uma atividade e um artefato (criar/atualizar/consultar);
- deve permitir a simulação de cenários em resposta a eventos internos eexternos;

Durante a definição do modelo SPIM, um conjunto de regras foi estabelecido para garantir a consistência do modelo. Todas as regras, bem como os conceitos por trás SPIM foram avaliados em uma série de entrevistas com experientes gerentes de projeto de software, cujos resultados podem ser vistos em [ROS08]. Neste sentido, foi desenvolvida anteriormente uma primeira versão de uma ferramenta que atua como um *add-in* para um software comercial de gerenciamento de projetos. Esta ferramenta, chamada *Software Planning Integrated Tool*- SPIT, foi desenvolvida para demonstrar a viabilidade dos conceitos propostos no modelo SPIM. Mais detalhes sobre a ferramenta SPIT são apresentados a seguir.

6.3 Implementação do Modelo

Partindo-se do modelo SPIM, realizou-se a implementação dos conceitos nele presentes. A implementação foi realizada em linguagem C#.net, utilizando o framework ‘.net’versão 4.5, com acesso ao banco de dados SQL Server 2008. Foram utilizados o

ambiente de programação Microsoft Visual Studio 2012. e a ferramenta de modelagem Jude Community versão 6.7.0. Esta seção descreve os demais detalhes de implementação.

A ferramenta *Software Planning Integrated Tool* - SPIT tem como objetivo oferecer algum tipo de suporte aos gestores no processo decisório de projetos de software através dos conceitos propostos pelo modelo SPIM e seu conjunto de regras. Para atingir este objetivo, este software foi desenvolvido de forma modular, permitindo que ele absorva novas funcionalidades de forma rápida e sem comprometer sua estrutura. Atualmente, o SPIT tem integrado ao seu ambiente os seguintes módulos:

- SPIM Validator: atua como um *add-in* para o Microsoft Project 2013 e executa as regras de validação do modelo SPIM sobre projetos de software;
- BackOffice: responsável pela gestão da informação requerida pelo modelo SPIM, como definição de papéis, tipos de atividades e produtos de trabalho associados. Esta informação é exportada para o Microsoft Project através de campos personalizados desta ferramenta comercial (dados que são posteriormente utilizados pelo módulo SPIM Validator);
- Workflow Integrator: módulo responsável por sincronizar as informações contidas nos fluxos de trabalho organizacionais com as informações contidas em um projeto de software específico;
- Compiler: uma ferramenta para modelagem, compilação, geração e verificação de redes de Petri. Este módulo usa internamente o formato PNML para representar uma rede de Petri.
- Simulator: Consiste em um ambiente para simulação e análise de redes de Petri.

As seções a seguir apresentam o detalhamento das funcionalidades da ferramenta SPIT, estando divididas em subseções por grupo de funcionalidades.

6.3.1 SPIM Validator

O módulo SPIM Validator age como um *add-in* para o software Microsoft Project 2013 [CHA10] e executa as regras de validação do modelo SPIM sobre projetos de software. Esta opção permite ao SPIT tirar proveito dos recursos que já foram implementados em conformidade com o modelo de integração proposto. No entanto, SPIM propor certos conceitos e limitações que não são implementadas por esta ferramenta comercial. Segundo Chatfield e Johnson [CHA10], este software comercial tem

campos extras que permitem o armazenamento de informações personalizadas para tarefas e recursos. Através deles, foi possível adicionar informações extras para as tarefas e recursos do projeto. No âmbito desta pesquisa, foi necessário adicionar campos personalizados para tarefas e recursos para realizar as validações especificadas na SPIM modelo.

Um trabalho anterior [CAL07] desenvolveu um metamodelo integração entre as classes do PMBOK e RUP (chamado PMBOK+RUP). Inicialmente, o estudo sobre a integração dos conceitos advindos do PMBOK e do RUP produziu um conjunto de 19 regras (cujos resultados foram publicados em [ROS08a]) para garantir a consistência do modelo. Conforme mencionado anteriormente, este estudo inicial permitiu o desenvolvimento da metodologia adotada para a integração de modelos de gestão de projetos com modelos de processos de desenvolvimento de software. Desta forma, 10 novas regras foram adicionadas àquelas desenvolvidas previamente (cujos resultados foram publicados em [ROS12b]). Estas restrições não puderam ser expressas através de diagramas devido às limitações na expressividade do diagrama de classes UML. Desta forma, aTabela 8 mostra o conjunto de restrições que foram definidas para o modelo integrado SPIM. Estas informações extras, propostas pelo SPIM, são armazenadas em um banco de dados externo. O módulo SPIM Validator foi atualizado para acessar um banco de dados Microsoft SQL Server 2008 Express [DEW08] através de *stored procedures*, devido à sua segurança e manutenabilidade.

Tabela 8: Conjunto de restrições do modelo integrado SPIM

#	**Restrições do modelo integrado SPIM**
1	Um programa deve possuir um diretor. Logo, um *stakeholder* que é diretor de um programa deve possuir um papel gerencial;
2	Um projeto deve ter apenas um *stakeholder*chave, ou seja, um recurso responsável por direcionar e fundamentar o projeto (atributo `isKeyStakeholder` da classe `ProjectStakeholder`);
3	Uma fase não pode ter ela mesma como predecessora ou antecessora;
4	As fases do projeto não podem ocorrer em paralelo, de maneira que uma fase deve ser totalmente finalizada antes de iniciar outra;
5	Uma atividade não pode ter ela mesma como predecessora ou antecessora;
6	O fluxo de atividades não pode resultar em um ciclo; por exemplo, a atividade A é pré-requisito para a atividade B e a atividade B é pré-requisito para a atividade A;
7	Uma mesma atividade não pode criar, modificar ou consultar um mesmo artefato. Para realizar tais operações devem ser criadas atividades distintas;
8	Uma atividade gerencial deve ter pelo menos um papel gerencial como um de seus papéis;
9	Uma atividade produtiva deve ter pelo menos um papel produtivo como um de seus papéis;
10	O *stakeholder* responsável por uma atividade gerencial deve possuir um papel

	gerencial;
11	O *stakeholder* responsável por uma atividade produtiva deve possuir um papel produtivo;
12	Cada atividade do projeto deve ser de responsabilidade de apenas um indivíduo, mesmo que muitas pessoas venham a trabalhar naquela atividade;
13	Uma atividade gerencial não pode produzir ou modificar um produto de trabalho produtivo, somente um produto de trabalho gerencial. Porém, esta atividade pode consultar um produto de trabalho produtivo;
14	Uma atividade produtiva não pode produzir ou modificar um produto de trabalho gerencial, somente um produto de trabalho produtivo. Porém, esta atividade pode consultar um produto de trabalho gerencial;
15	Para cada associação entre um papel e uma atividade (representado pela classe ActivityStakeholderWork) deve haver também uma associação dessa atividade com um *stakeholder* capaz de desempenhar aquele papel;
16	Uma atividade somente pode atualizar ou consultar um produto de trabalho que já tenha sido criado por uma atividade antecessora;
17	Uma atividade pode ou não ter baselines. Se tiver, a atividade original deve ter o atributo `IsBaseline=false` e todas as outras atividades (relacionadas via a associação `Baselines`) devem manter `IsBaseline` como `true`.
18	Na classe associativa `ActivityStakeholderWork`, o *stakeholder* associado deve possuir o papel que se está associando à atividade.
19	O papel do *stakeholder* envolvido deve ser compatível com o tipo de atividade.
20	O produto de trabalho envolvido deve ser compatível com o tipo de atividade.
21	Uma disciplina modela apenas uma coleção de atividades produtivas que contém um objetivo comum;
22	Os produtos podem ser documentados utilizando várias linguagens. Entretanto, um produto específico deve ser documentado somente com uma linguagem específica;
23	Uma ferramenta gerencial não pode ser relacionada com atividades que produzem ou modificam um produto de trabalho produtivo, apenas com um produto de trabalho gerencial. No entanto, pode consultar um produto de trabalho produtivo;
24	Uma ferramenta produtiva não pode estar relacionada com atividades que produzem ou modificam um produto de trabalho gerencial, apenas com um produto de trabalho produtivo. No entanto, pode consultar um produto de trabalho gerencial;
25	Um conjunto de equipes da organização deve ser uma coleção coesa de equipes. Assim, este conjunto não pode conter as equipes que têm apenas funções gerenciais. Neste caso, não haveria produtos de trabalho produtivos;
26	Um conjunto de equipes da organização deve ser uma coleção coesa de equipes. Assim, este conjunto não pode conter as equipes que têm apenas funções produtivas. Neste caso, não haveria produtos de trabalho gerenciais;
27	Os produtos de trabalho produtivos devem ser documentados com uma linguagem dita produtiva;
28	Os produtos de trabalho produtivos devem ser documentados com uma linguagem dita gerencial;
29	A duração das fases é calculada somando-se o tempo de suas atividades associadas;
30	Os ciclos de um projeto de software não devem prosseguir em paralelo;

Este conjunto de restrições está subdividido duas categorias: alertas e impeditivas. As restrições categorizadas como sendo alertas atuam como instrumento informativo para o gestor de projetos (caberá ao gestor acatar ou não às informações repassadas pelo

modelo SPIM). As restrições impeditivas, entretanto, obrigam o gestor a corrigir o plano de projeto. Desta forma, as seguintes restrições são consideradas como impeditivas: #3, #4, #5, #6, #8, #9, #10, #11, #13, #14, #17, #19, #20, #29 e #30. As outras restrições estão categorizadas como sendo alertas do modelo SPIM.Cabe salientar, ainda, que o software Microsoft Project oferece suporte para as seguintes restrições propostas no modelo SPIM: #3, #4, #5, #6, #17, #29 e #30.

6.3.2 BackOffice

O módulo de Backoffice é responsável por registrar as informações exigidas pelo SPIM modelo, tais como definições de papel, tipos de atividades e produtos de trabalho associados, em um banco de dados externo. Esta informação é exportada para o software Microsoft Project através de campos personalizados para serem usados pelo módulo SPIM Validator. Para realizar essa tarefa, o módulo de BackOffice fornece uma interface (ver Figura 39) que permite aos gerentes de projeto associar um projeto de software registrado na base de dados do modelo SPIT com um arquivo usado pelo Microsoft Project (.mpp).

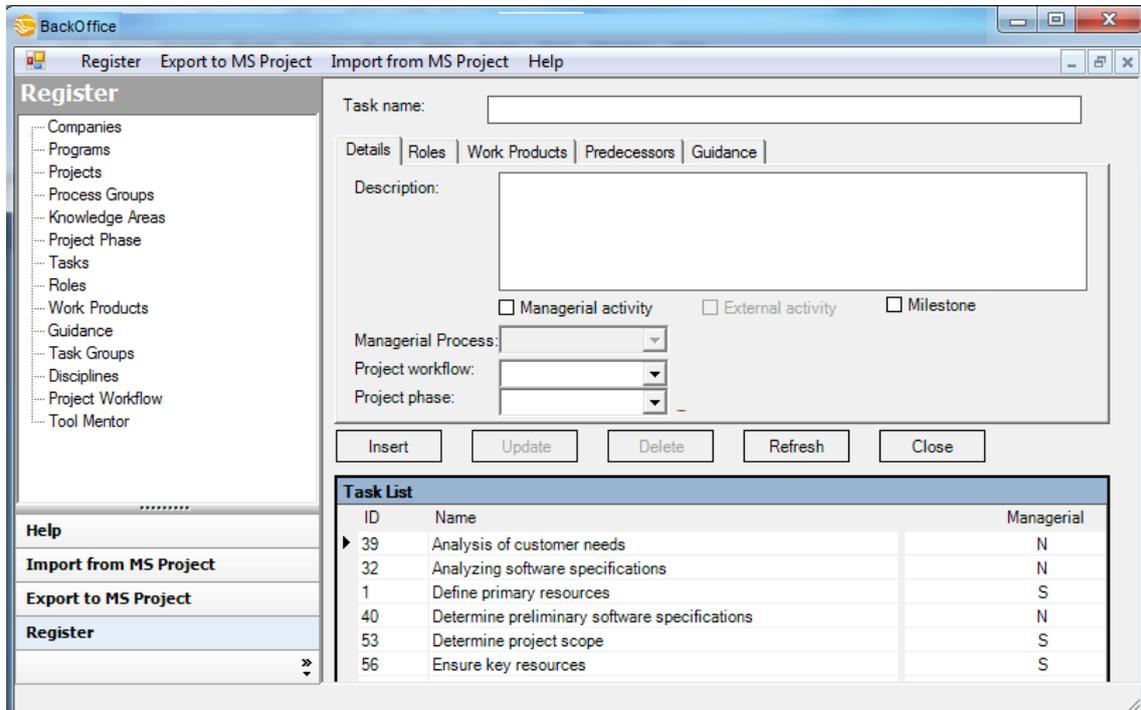



Figura 39. Tela do BackOffice

O módulo de BackOffice oferece duas formas de exportar as informações contidas em seu banco de dados para o Microsoft Project:

a) através de modelos de projeto; e

b) através individuais campos personalizados.

As informações necessárias para executar a validação de acordo com as restrições impostas pelo modelo integrado SPIM devem estar em campos personalizados do Microsoft Project. Uma maneira de exportar informações do SPIT para a ferramenta comercial é realizado através de um modelo de projeto. Assim, um conjunto de tarefas e as suas associações serão exportados para o software comercial de uma só vez. Também é possível selecionar individualmente os campos personalizados que serão exportados para um projeto específico. Assim, o módulo de BackOffice permite exportar informações sobre os papéis, produtos de trabalho, processos gerenciais do PMBOK, guias e fluxos de trabalho do RUP e do OPEN.

6.3.3 Workflow Integrator

O módulo Workflow Integrator é responsável por sincronizar as informações contidas nos fluxos de trabalho organizacionais com as atividades contidas em um projeto de software específico. Atualmente, diferentes tecnologias e soluções podem ser encontradas para o desenvolvimento de fluxos de trabalho. O *Windows Workflow Foundation*(WWF) fornece o *Visual Studio Workflow Designer 2010*, que permite aos usuários criar seus próprios fluxos de trabalho personalizados [MYE07]. Desta forma, o *Windows Workflow Foundation* é uma estrutura extensível para o desenvolvimento de soluções de fluxo de trabalho na plataforma Windows. As soluções baseadas no WWF são construídas com componentes interconectados que têm o suporte do código do Microsoft '.NET' e são executados em um aplicativo *host*. Ainda, o WWF fornece um mecanismo de *workflow*, uma API gerenciada pelo '.NET'*Framework*, serviços de tempo de execução e um *designer* e depurador visual integrado ao Microsoft Visual Studio 2012 [MYE07].

Basicamente, workflows criados através deste *framework* podem ser classificados em dois tipos:

- *Sequential workflow*: o fluxo de execução das atividades ocorre passo-a-passo, sendo possível ainda o uso de desvios condicionais dentro da lógica que se está elaborando. Seu uso é recomendado na modelagem de processos mais simplificados e sem intervenção humana, sendo equivalentes a rotinas escritas de maneira procedural dentro de uma linguagem de programação;

- *Flowchart workflow*: permite a modelagem de um processo num padrão gráfico similar a um diagrama de atividades UML (vide Figura 40). São úteis na representação de processos com uma estrutura sequencial, mas que também contam com *loops* que desviam o fluxo de execução para estados anteriores. Diferentemente de um *Sequential workflow*, este tipo de *workflow* é recomendável na modelagem de processos que também contem com interações humanas. Em um *Flowchart workflow* as diversas atividades que fazem parte de um processo são agrupadas dentro de uma atividade do tipo *Flowchart*. Nesta tese, desenvolveram-se fluxos de trabalho deste tipo.

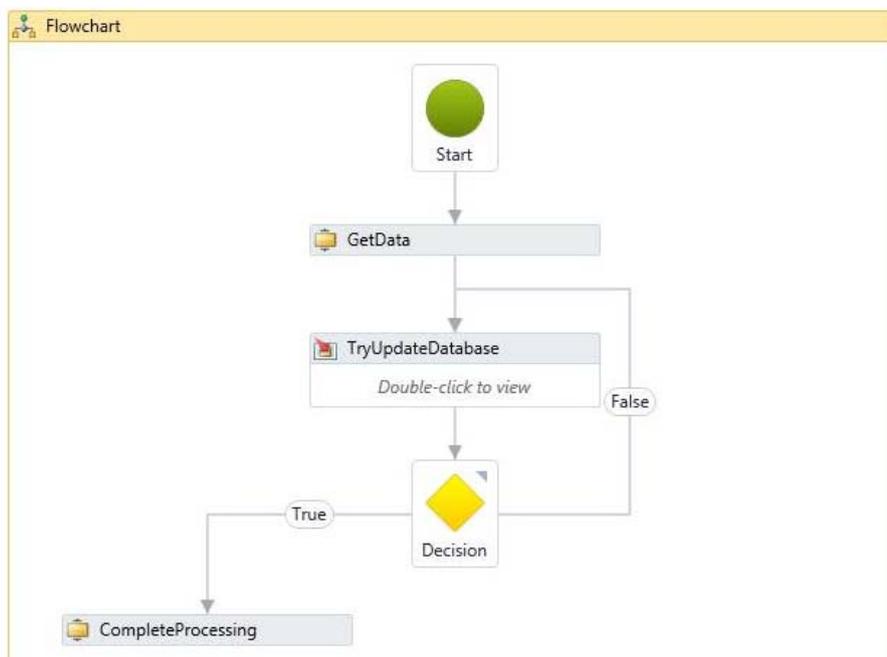



Figura 40. Exemplo de *Flowchart workflow*

As interações entre as atividades que compõem um *workflow* envolverão, em grande parte dos casos, o fluxo de informações de um ponto a outro do processo em questão. Assim, procurando oferecer suporte ao intercâmbio de dados entre as partes que constituem um *workflow*, a tecnologia *Workflow Foundation* disponibiliza recursos como variáveis, argumentos e expressões, utilizando para isto dos mesmos conceitos presentes numa linguagem de programação convencional [MYE07].

Devido a natureza distribuída de um processo de negócio, que considera informações provenientes de vários departamentos em uma organização, um fluxo de trabalho pode ser implementado como sendo uma aplicação distribuída. Para suportar esta funcionalidade, o uso de serviços web (*web services*) foi escolhido como a plataforma para acessar aplicações distribuídas de *workflow*. *Web services* foram

projetados para abstrair como os aplicativos se comunicam uns com os outros. Estes serviços permitem a separação da lógica de negócio com o código do aplicativo cliente. Então, utiliza-se *web services* para expor pontos de interação, modelados dentro do fluxo de trabalho, como métodos de serviço da web.

Para melhor ilustrar estes recursos propostos no SPIT, será utilizado um projeto de desenvolvimento de software fictício. Considerando o modelo SPIM, em conjunto com as funcionalidades desenvolvidas para o SPIT, o gestor de projetos deve iniciar o planejamento realizando a configuração inicial do projeto (tais como a definir papéis e produtos de trabalho) através do módulo de *BackOffice*. Uma vez cadastrada no banco de dados do SPIT, esse tipo de informação deve ser exportada para um arquivo que possa ser utilizada pelo Microsoft Project 2013. Cabe relembrar que, durante o planejamento e execução do projeto de software, o gerente pode utilizar o módulo SPIM Validator potencializar as funcionalidades oferecidas pelo software comercial de gestão erealizar validações no projeto considerando as regras SPIM. Para exemplificar esta funcionalidade, o SPIM Validador pode informar que uma atividade produtiva (tal como, desenvolver um componente de software) foi erroneamente associada a uma função gerencial (gerente de sistemas, por exemplo). Considere, também,um cenário para a aquisição de um novo hardware (atividade realizada pelo setor administrativo da empresa), nesta situação o SPIT utiliza o módulo Workflow para consultar o fluxo organizacional correspondentesobreos prazos atualizados para realizar essa aquisição.

6.3.4 Compiler

O módulo Compiler é uma ferramenta para modelagem, compilação, geração e verificação de redes de Petri. O módulo Compiler pode fazer a tradução (em ambos os sentidos) entre as informações contidas na estrutura de trabalho do projeto (WBS) e nas redes de Petri. Ele usa a linguagem*Petri Net Markup Language – PNML* [JUN00] como um formato de intercâmbio baseado em XML para redes de Petri. A PNML foi proposta para definir as características típicas de todas as redes de Petri.

Segundo Murata [MU89], a maioria dos grupos de pesquisa sobre redes de Petri têm as suas próprias ferramentas para auxiliar de desenho, análise e/ou simulação de várias aplicações. Cada um deles têm suas características e pontos fortes específicos. Mas, esta variedade de ferramentas implica a persistência de vários formatos de arquivo. Cada ferramenta tem o seu próprio formato de arquivo, de modo que o usuário que

pretende combinar diferentes ferramentas de redes de Petri não pode usar o mesmo arquivo para analisar, simular ou verificar a mesma rede em ferramentas diferentes. De acordo com Jungel et al. [JUN00], a melhor maneira de resolver este problema é utilizar um formato de intercâmbio que pode ser lido por cada uma das ferramentas utilizadas. Estas ferramentas, no entanto, precisam representar as suas características específicas neste formato de intercâmbio. O formato de intercâmbio permite a interação de redes de tipos diferentes, mas compatíveis redes de Petri.

A arquitetura utilizada para o desenvolvimento do módulo Compiler, no entanto, permite a adição de novos formatos de saída (detalhes sobre esta arquitetura foram publicados em [ROS12c]). Conforme ilustrado na Figura 41, este módulo está divido em três partes:

- Parser: contém as classes que compõem o analisador para arquivos com extensão‘.pnml’ e ‘.mpp’;
- Translator: contém as classes que compõem a representação C#.net da WBS e da rede de Petri no SPIT;.
- Writer: contém as classes responsáveis por escrever os arquivos de saída.

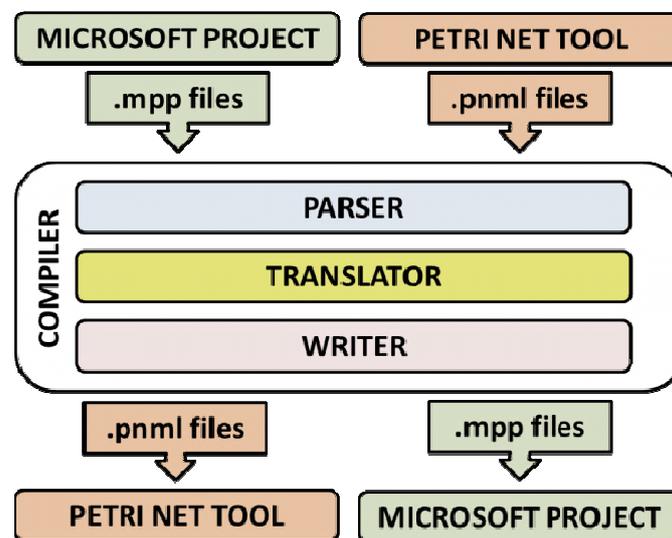



Figura 41. Formatos de entrada e saída do módulo Compiler [ROS12c]

O arquivo de entrada é lido e as informações sobre a rede de Petri e seus elementos são extraídas para criação de um objeto `PetriNet`, que é mantido em memória. Este objeto encapsula todos os elementos da rede, tornando possível que a partir dele sejam geradas as saídas para os outros formatos.

6.3.4.1 Arquitetura do Módulo Compiler

O módulo Compiler divide-se basicamente em duas partes: o pacote `core`, que contém as classes relacionadas aos processos de compilação e geração de arquivos; e o pacote `ui`, que trata das interfaces com o usuário no ambiente SPIT. Este módulo apresenta, desta forma, a seguinte estrutura de pacotes, ilustrada na Figura 14:

- integratedModel.compiler.core.entities: contém as classes que compõem a representação em C#.net de um rede de Petri.
- integratedModel.compiler.core.io.pnml: contém as classes responsáveis por realizar a gravação dos arquivos de saída no formato PNML.
- integratedModel.compiler.core.io.msProject: contém as classes responsáveis por realizar a gravação dos arquivos de saída no formato para o Microsoft Project.
- integratedModel.compiler.core.parser.pnml: contém as classes que compõem o parser de arquivos no formato PNML.
- integratedModel.compiler.core.parser.msProject: contém as classes que compõem o parser de arquivos no formato para o Microsoft Project.
- integratedModel.compiler.core.util: Classes de uso comum a todo o módulo.

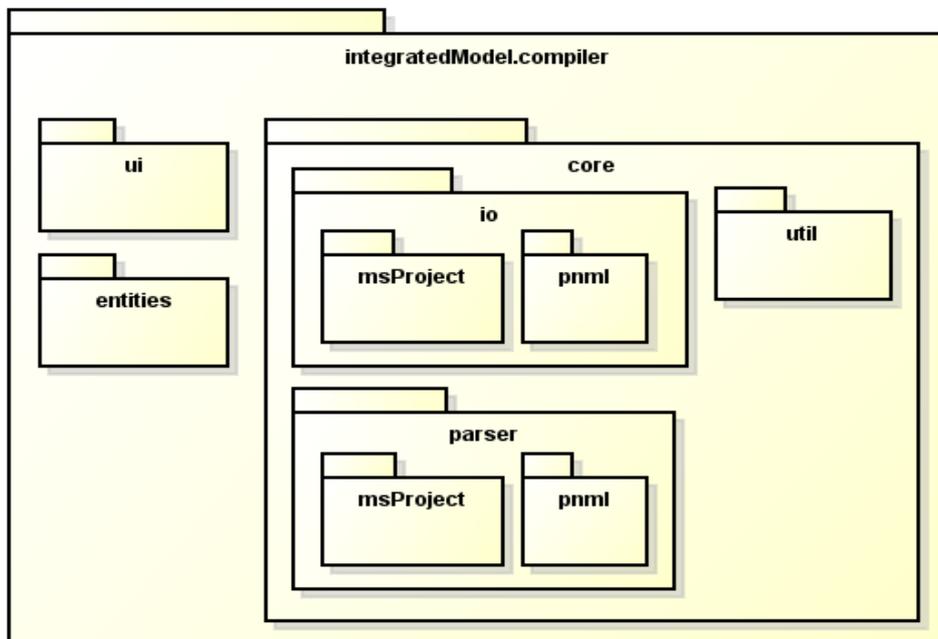



Figura 42. Diagrama de pacotes do MóduloCompiler

A seguir será detalhado o pacote `entities`, cujos elementos representam o modelo definido no compilador de uma rede de Petri estocástica.

6.3.4.2 Pacote Entities

Contém as classes que representam as entidades básicas do sistema. Assim, cada classe representa uma categoria de elementos que compõem uma rede de Petri gerada pelo SPIT. As classes que compõem este pacote, ilustradas na Figura 43, são listadas a seguir:

- PetriNet: representa uma rede de Petri com todos os seus possíveis elementos;
- Node: classe com elementos comuns às classes `Transition` e `Place`;
- Place: representa um lugar de uma rede de Petri;
- Transition: representa uma transição de uma rede de Petri;
- Arc: representa um arco de uma rede de Petri;
- Coordinate: representa um conjunto de coordenadas cartesianas x e y.

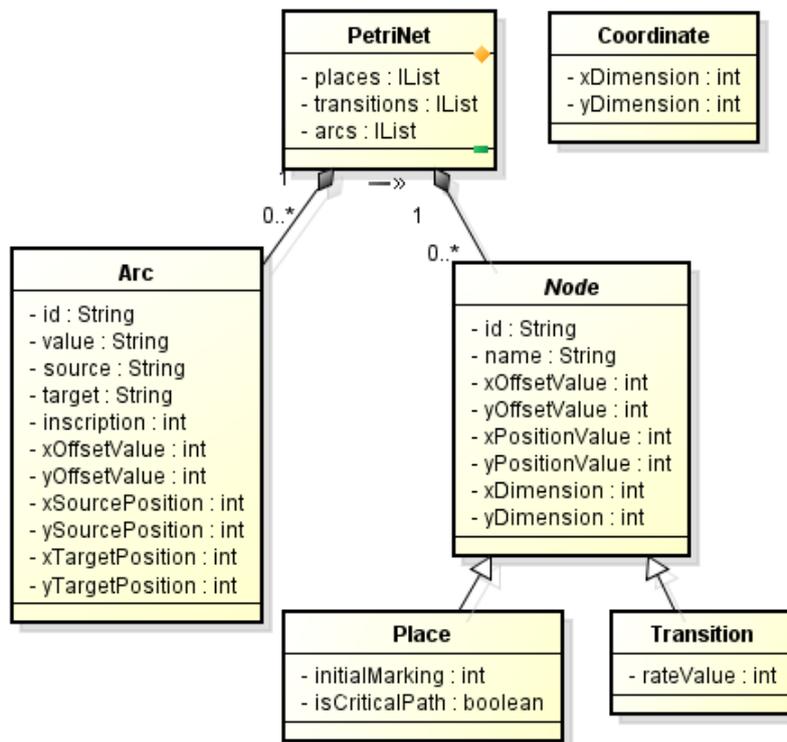



Figura 43. Diagrama de Classes do pacote entities

Conforme observado neste diagrama, a classe `PetriNet` é uma composição de arcos (classe `Arc`) e nodos (classe `Node`). Cada arco deve conter, basicamente, um identificador único, ponto de origem, ponto de destino e informações sobre o tamanho e o posicionamento. A classe `Node`contém informações de identificação, posicionamento e tamanho. Esta classe abstrata é herdada por duas classes concretas:`Place` e `Transition`. A classe `Place` contém informações sobre o número de tokens e se faz

parte do caminho crítico. A classe `Transition` contém o valor sobre o tempo de duração desta transição.

As subseções a seguir apresentam as características sobre o *parsing* de arquivos de entrada e a geração de arquivos de saída.

6.3.4.3 Pacote Parser

Um *parser* foi desenvolvido para cada um dos formatos de entrada: um para leitura de arquivos importados do Microsoft Project (arquivo '.mpp') e um para o formato PNML (arquivo '.pnml'). Ambos apresentam duas etapas: a primeira consiste em agrupar os elementos de acordo com seu tipo (tal como: lugares, transições e arcos) e a segunda consiste em extrair as informações de cada um dos elementos.

As classes envolvidas nos processos de leitura e *parsing* de arquivos estão encapsuladas no pacote `integratedModel.compiler.core.parser`(vide Figura 42). Todos os *parsers* criados devem implementar a interface `IParser` existente neste pacote. Utilizou-se o padrão de projeto chamado *Factory* (através da classe `ParserFactory`) para instanciar objetos deste pacote. O padrão *Factory* fornece uma abstração (ou uma interface) e permite que cada subclasse possa decidir qual classe ou método deve ser chamado ou instanciado, com base nas condições ou parâmetros relatados. Logo, a classe `ParserFactory` é responsável por criar instâncias de classes que implementam a classe `IParser`, de acordo com a extensão do arquivo de entrada ('.mpp' ou '.pnml').

Entretanto, nenhum dos *parsers* desenvolvidos implementa algum tipo de validação léxica e estrutural. Os valores definidos pelos usuários em ambos os arquivos de entrada, bem como no ambiente de modelagem SPIT, são tratados como um valor textual, sem valor semântico. Este tipo de validação e tratamento fazem parte das propostas para trabalhos futuros nesta pesquisa.

6.3.4.3.1 Parsing de arquivos MPP

As classes utilizadas para o parsing de um arquivo ‘.mpp’ são agrupadas no pacote `integratedModel.compiler.core.parser.msProject`. Incialmente, os dados do arquivo de entrada são agrupados de acordo com seus tipos (recursos e tarefas) e são passados para as classes responsáveis por traduzir os elementos da WBS em objetos C#. Para utilizar os recursos de um aplicativo do Microsoft Office, tal como o Microsoft

Project, deve-se usar componente chamado *Primary Interop Assembly* (PIA). Em seguida, a extração de informação ocorre de forma sequencial, de acordo com a estrutura do arquivo de entrada. Assim, o *parser* cria um objeto que representa a rede de Petri (classe `petriNet`) e percorre todas as tarefas do projeto. Para cada tarefa deste projeto é criado um elemento de lugar desta nova rede de Petri. O elemento local armazena as informações sobre a identificação e descrição da tarefa, se esta faz parte do caminho crítico do projeto e outros dados (como posição e dimensão) necessários para representá-la graficamente na ferramenta SPIT. Durante este processo, o algoritmo verifica se essa tarefa é antecessora de outra tarefa. Se isso acontecer, o *parser* irá criar os elementos de arco e transição para ligar estes dois elementos lugar. O arco liga lugares com transições de acordo com a relação de precedência entre as tarefas do projeto (ver Figura 44).

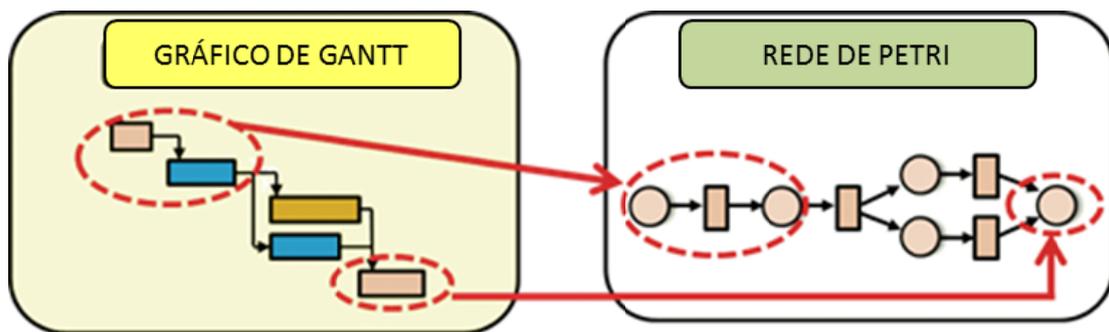



Figura 44. Tradução dos dados do gráfico de Gantt em elementos da rede de Petri – adaptado de [ROS12c]

A Figura 45 mostra a interface do SPIT para gerar redes de Petri a partir da WBS. Na parte superior desta tela, o usuário deve selecionar um dos projetos de software que estão sendo visualizados no Microsoft Project e clicar no botão “Generate Petri Net”. As mensagens sobre o resultado desta importação pode ser observado no campo “Results”. A parte esquerda da tela mostra ao usuário a estrutura da rede de Petri gerada, informando os lugares, transições e arcos que fazem parte desta rede. Ao selecionar um item da rede (um determinado lugar, por exemplo), esta interface apresenta os detalhes deste item na parte inferior da tela (informações sobre o identificador do item, seu nome, se pertence ao caminho crítico, entre outras). Finalmente, a parte esquerda da tela mostra ao usuário a rede de Petri gerada. Aqueles lugares que pertencem ao caminho crítico estão em vermelho, para auxiliar na identificação pelo gestor do projeto.

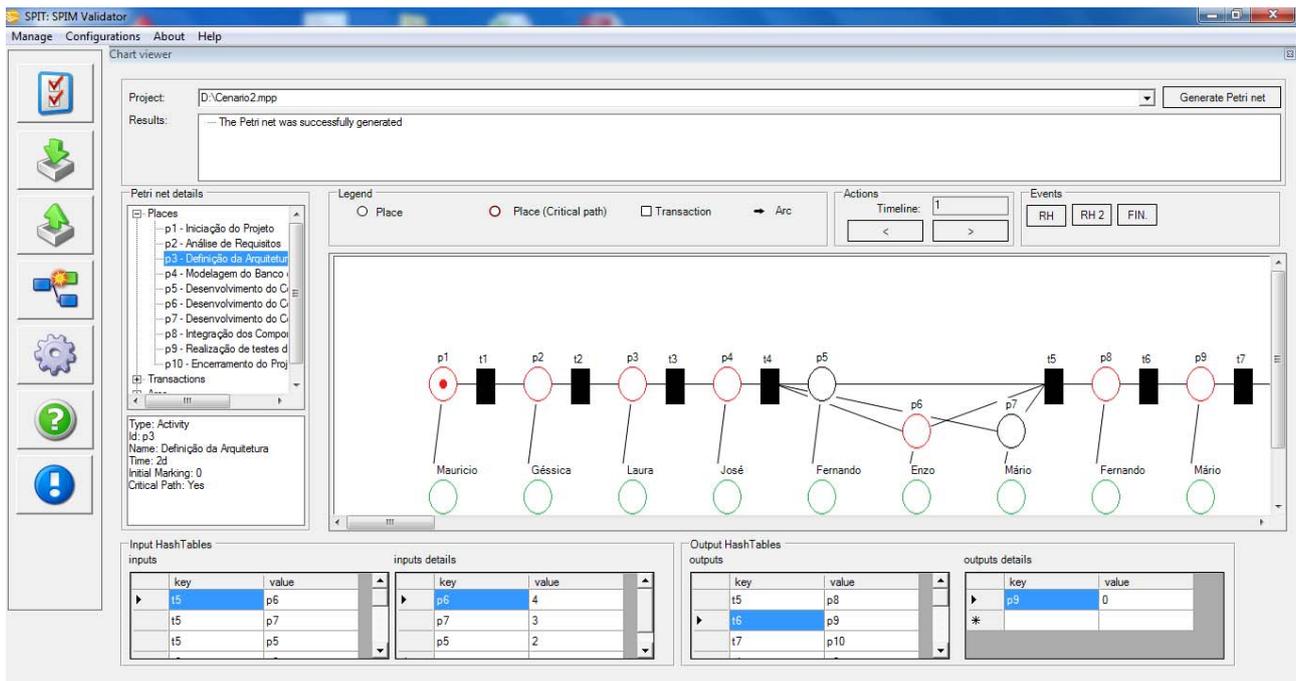



Figura 45. Geração de redes de Petri no SPIT

6.3.4.3.2 Parsing de arquivos PNML

As classes utilizadas para o parsing de um arquivo com extensão '.pnml' são agrupadas no pacote `integratedModel.compiler.core.parser.pnml`. Conforme salientado anteriormente, PNML é um formato de intercâmbio, baseado em XML, para representar redes de Petri. De maneira análoga à utilizada na leitura de arquivos com extensão '.mpp', os elementos da rede de Petri representados no arquivo de entrada são agrupados de acordo com seus tipos (local, transição e arco) e, em seguida, são passados como parâmetros para as classes responsáveis por traduzir os elementos PNML em objetos C#. No entanto, esta tradução apresenta algumas peculiaridades, devido a possibilidade do arquivo de entrada ser gerado por outra ferramenta. Para evitar inconsistências no processo de tradução, definimos duas regras básicas para validar o conteúdo dos arquivos PNML de entrada:

a) verificar se o elemento ID representado no documento PNML é necessariamente inteiro, e
b) verificar se o elemento de nome representado no documento PNML começa com uma letra.

Visando um melhor desempenho no processo de conversão, essas validações são implementadas usando o mecanismo de expressão regular oferecido pelo *'.Net'*

Framework. Durante este processo de validação, os nomes ou identificadores inválidos são substituídos com IDs e nomes válidos. Estas modificações devem ser mantidas para que, durante a leitura de elementos que possuem referências a outros elementos da rede, como por exemplo os arcos, não sejam introduzidas inconsistências no modelo gerado.

A Figura 46 apresenta uma RdP utilizada para exemplificar o funcionamento do *parsing* de arquivos PNML desenvolvido para o protótipo SPIT.

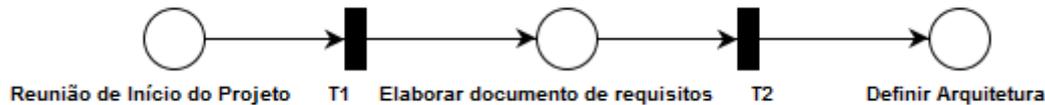



Figura 46. Rede de Petri para um projeto de software fictício

Após a validação do arquivo de entrada, que contém a RdP no formato PNML, o arquivo é analisado. Os lugares descritos no código PNML dentro das etiquetas `<place id = "rótulo do lugar">` e `</place>`, são os primeiros componentes analisados, pois pela identificação do valor dos seus nomes (encontrado dentro das etiquetas `<value>` e `</value>`) é possível computar as atividades que farão parte da WBS do projeto de software.

Na Figura 47, é ilustrado o trecho do código PNML que descreve os lugares da RdP apresentada naFigura 46. Observa-se que os lugares “Reunião de Início do Projeto”, “Elaborar documento de requisitos” e “Definir Arquitetura” representam as atividades do projeto, pois os valores atribuídos a esses lugares representam o tempo de execução destas (1 unidade de tempo para a primeira atividade, 3unidades de tempo para a segunda atividade e 2 unidades de tempo para a terceira atividade). O número atividades é calculado, dessa forma, de acordo com a quantidade de lugares existentes no modelo da RdP. As informações contidas na tag `<graphics>` são utilizadas para representar graficamente estes objetos, informando sua posição e dimensão na tela.

```
<place id="p1">
  <name>
    <text>Reunião de Início do Projeto</text>
    <value>1<value>
  </name>
  <graphics>
    <position x="100" y="100" />
    <dimension x="30" y="30" />
  </graphics>
</place>
<place id="p2">
  <name>
    <text>Elaborar documento de requisitos</text>
    <value>3</value>
  </name>
  <graphics>
    <position x="200" y="100" />
    <dimension x="30" y="30" />
  </graphics>
</place>
<place id="p3">
  <name>
    <text>Definir Arquitetura</text>
    <value>2</value>
  </name>
  <graphics>
    <position x="300" y="150" />
    <dimension x="30" y="30" />
  </graphics>
</place>
```

Figura 47. Trecho do código PNML onde estão descritos os lugares da RdP

Em seguida, são analisados as transições da RdP (descritos no código PNML dentro das etiquetas `<transition ...>` e `</transition>`). Estes elementos auxiliam na identificação sobre a existência de relações de dependência entre as atividades do projeto. Conforme ilustrado na Figura 48, a identificação de cada transição é representado no código PNML pelo trecho `<text>”nome da transição”</text>`. As informações contidas na tag `<graphics>` são utilizadas para representar graficamente estes objetos, informando sua posição e dimensão na tela.

```
<transition id="t1">
  <name>
    <text>t1</text>
  </name>
  <graphics>
    <position x="150" y="100" />
    <dimension x="30" y="30" />
  </graphics>
</transition>
<transition id="t2">
  <name>
    <text>t2</text>
  </name>
  <graphics>
    <position x="350" y="100" />
    <dimension x="30" y="30" />
  </graphics>
</transition>
<transition id="t3">
  <name>
    <text>t3</text>
  </name>
  <graphics>
    <position x="350" y="200" />
    <dimension x="30" y="30" />
  </graphics>
</transition>
```

Figura 48. Trecho do código PNML onde estão descritos as transições da RdP

Conforme observado na Figura 49, os arcos descritos no código PNML nas etiquetas `<arc...>` e `</arc>`, são utilizados para obter a ordem de precedência entre as atividades. Assim, através da descrição dos arcos é possível identificaas atividades dependentes e as atividades precedentes do projeto. As atividades precedentes são localizadas nos arcos que representam a ligação dos lugares às transições, identificados no código PNML pelo trecho `source="rótulo do lugar" etarget="rótulo da transição"`.As atividades dependentes são localizadas nos arcos que realizam a ligação das transições aos lugares, identificados no trecho de código PNML `source="rótulo da transição" e target="rótulo do lugar"`.

```
<arc id="a1" source="p1" target="t1">
  <graphics />
</arc>
<arc id="a2" source="t1" target="p2">
  <graphics />
</arc>
<arc id="a3" source="t1" target="p3">
  <graphics />
</arc>
```

Figura 49. Trecho do código PNML onde estão descritos os arcos da RdP

Analisando, identificando e processando esses componentes, podem-se gerar os elementos da WBS para este projeto de software em memória. Após, é possível exportar estas informações para um arquivo com extensão'.mpp'. A próxima subseção apresenta maiores detalhes sobre o processo de exportação das informações pelo SPIT.

6.3.4.4 Pacote IO

A partir do momento em que o processo de *parsing* da entrada se encerra, tem-se um objeto `petriNet` pronto para ser persistido, seja no formato de arquivo .'mpp' ou '.pnml'. As classes envolvidas na gravação de arquivos estão encapsuladas no pacote `integratedModel.compiler.core.io`. Estas classes devem implementar a interface `IWriter` existente neste pacote. A instanciação dessas classes é feita de maneira análoga à utilizada na etapa de *parsing* dos arquivos de entrada, através da classe `WriterFactory`. A classe `WriterFactory` é responsável por gerar instâncias das classes de gravação de arquivos, de acordo com a extensão do arquivo de saída desejado ('.mpp' ou '.pnml').

Esta abordagem trouxe diversas facilidades para o processo de geração dos arquivos de saída, pois pode-se especificar diferentes formatos de arquivos de saída a partir de um único modelo mantido na memória. O *writer*para documentoscom extensão '.pnml' foi desenvolvido utilizando o *XML DOM API* do *Framework '.Net'* enquanto que o *writer* para documentos com extensão '.mpp' foi utilizado com apoio da biblioteca *Microsoft Project Object Library*.

6.3.4.4.1 Geração da Rede de Petri

A geração da rede de Petri que representa as informações contidas na WBS de um projeto de software foi apresentado na seção "5.3 Mecanismo de Mapeamento entre o

Diagrama de Rede e a Rede de Petri”. Resumidamente, a RdP é gerada através das seguintes regras:

1. Cada atividade do projeto é transcrita como um lugar, uma transição e um arco de entrada ligando o lugar à transição.
2. O início do projeto é definido pela alocação de uma ficha em um lugar da RdP.
3. O ponto de convergência deatividades executadas em paraleloé representado por uma transição e dois ou mais lugares de entrada.
4. Um ponto no projeto onde uma atividade é predecessora de duas ou mais atividades é representado através de uma transição imediata (considera-se nesta tese que o tempo gasto para a realização da atividade está registrado no lugar que corresponde a esta atividade). Assim, ligadas ao lugar de saída há duas ou mais transições imediatas, responsáveis pela representação de cada fluxo a ser seguido.
5. Cada agrupamento de atividades, tal como uma fase, não são considerados na geração da rede de Petri.
6. O recurso alocadoparauma atividade deve ser representado com um objeto em um lugar de recurso, o qual deve ser ligado até a transição que representa a atividade por um arco restaurador.

6.3.4.4.2 Geração da WBS

Através das informações referentes aos projetos de software, que estão contidos na rede de Petri (contidas em lugares, arcos e transições), a WBS é gerada através das seguintes regras:

1. Cada lugar da rede de Petri é transcrita como uma atividade da WBS. As informações sobre o tempo das atividades também são extraídas dos lugares da RdP.
2. O ponto de início do projeto é identificado pelo lugar que contém uma ficha na RdP.
3. As atividades que dependem da execução de outras atividades (predecessoras) são transcritas

4. Um ponto no fluxo da rede onde diferentes fluxos paralelos convergem em um único fluxo é representado por uma atividade e n atividades predecessoras.
5. A classe de um lugar que corresponde a um recurso executor é transcrito para um objeto de recurso na WBS.

6.3.5 Simulator

O módulo Simulator implementa o simulador de redes de Petri do software SPIT. Objetivando a otimização de desempenho deste módulo, as redes de Petri são representadas internamente considerando apenas as informações relevantes para a simulação. Desta forma, o modelo de representação de RdP para o módulo Simulator foi desenvolvido de acordo com os seguintes critérios:

- Não-redundância:as informações redundantes na representação da RdP foram excluídas. Ainda, o modelo de representação da RdP é compacto para evitar o desperdício dememória.
- Acesso rápido: o simulador precisa obter rapidamente as informações que procura, de modo que o simulador define uma janela de tempo para limitar suas buscas.
- Relevância:apenas as informações necessárias para a simulação são mantidas. Por exemplo, as informações gráficas (posicionamento e tamanho) foram eliminadas.

Com base nestes critérios, o modelo de representação da RdP para o módulo Simulator foi desenvolvido considerando a utilização de tabelas de dispersão (do inglês *hash*). A tabela *hash* é uma estrutura de dados que associa chaves de pesquisa a valores. As tabelas hashing são estruturas de dados que têm alto desempenho na realização de buscas, pois utilizam um algoritmo de codificação que permite que os dados sejam encontrados rapidamente, independente da quantidade de dados armazenados (algoritmo de complexidade constante ou $O(1)$). A sua vantagem com relação às outras estruturas associativas (como um vetor simples) passa a ser maior conforme a quantidade de dados aumenta.

Uma rede de Petri contém informações sobre são os seus lugares, transições e arcos. Estas informações, entretanto,representam excessiva redundância sob o ponto de vista da simulação. Observa-se que toda a estrutura da rede pode ser recuperada a partir

de seu conjunto de arcos. Cabe relembrar que os arcos são um conjunto de pares (origem, destino). Desta forma, o conjunto de arcos foi dividido em dois conjuntos menores: um contendo apenas os arcos que tem os lugares como ponto de origem e outro contendo os arcos que tem as transições como ponto de origem.

Para que a RdP seja obtida, faz-se necessário seguir os passos a seguir:

1. Para cada arco no conjunto que parte de lugares, o primeiro elemento do par é adicionado ao conjunto de lugares da rede e o segundo elemento ao seu conjunto de transições;
2. Para cada arco no conjunto que parte de transições, se o primeiro elemento do par ainda não estiver no conjunto de transições da rede, ele é adicionado; se o segundo elemento ainda não fizer parte do conjunto de lugares, ele é adicionado;
3. Todos os arcos são adicionados ao conjunto de arcos da rede.

Pode-se verificar, desta forma, que a RdP pode ser recuperada a partir do seu conjunto de arcos.Assim, o simulador desenvolvido considera apenas os arcos para representar a estrutura da rede. Duas tabelas são preenchidas, uma para os arcos que entram em uma transição (tabela de entrada) e outra para os que partem da transição (tabela de saída). Por questões de otimização, conforme observado na Figura 50, as tabelas de entrada e de saída foram estruturadas hierarquicamente. As tabelas de entrada e saída são compostas por uma tabela principal e várias tabelas secundárias. Cada tabela principal contém os identificadores dastransições como chave. Assim, cada chave está associada uma tabela secundária. As tabelas secundárias contêm como chave osidentificadores dos lugares que se conectam a transição relacionada na tabela principal através deum arco. Os valores nesta tabela secundária são os pesos dos arcos.

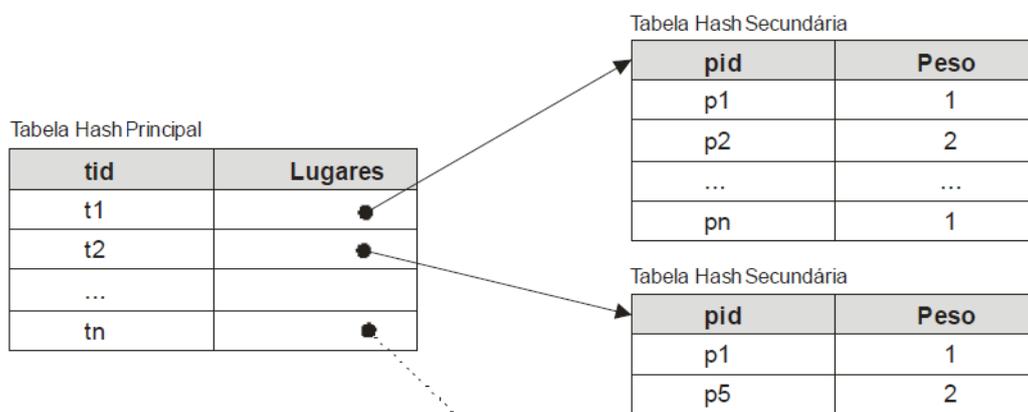



Figura 50. Hierarquia em que foram estruturadas as tabelas

Esta estrutura mostra-se apropriada uma vez que o simulador precisa obter todos os arcos para uma determinada transição e, em seguida, obter o peso de acordo com cada lugar de entrada ou de saída.As informações das tabelas de entrada e saída são carregadas apenas no início do processo de simulação, mas são necessárias durante toda a simulação. As informações sobre marcação da rede e estado das transições, entretanto, se modificam constantemente durante o processo de simulação.

Objetivando minimizar os recursos mantidos em memória, consideram-se apenas os lugares que possuem *tokens* para registrar o estado de marcação da rede. Assim, os lugares que não possuem nenhum *token* podem ser eliminados da tabela de marcações. No momento em que o simulador precisar consultar a marcação de um lugar, se o lugar não for encontrado na tabela de marcações, ele deduz que o lugar procurado não possui *token*. Este mesmo critério é utilizadona tabela de transições habilitadas, ou seja, apenas as transições que estão habilitadas são mantidas na memória.

Além de eliminar-se da tabela de marcação os lugares que não possuem *tokens*, estes lugares são também ignorados no processo de avaliação das transições – que calcula se cada transição está habilitada ou não. Quando o simulador procura por transições a serem habilitadas, ele procura apenas por aquelas que possuem alguma marcação em suas pré-condições.O algoritmo de simulação das redes pode ser descrito como a sequência de passos a seguir:

1. analisar as transições com a marcação atual;
2. habilitar aquelas que devem ser habilitadas e desabilitar as demais;
3. se não houver transições habilitadas, o processo de simulaçãoé finalizado;
4. identificar qual(is) transição(ões) deve(m) ser disparada(s);
5. disparar a transição escolhida, fazendo as mudanças das marcações dos lugares;
6. enquanto houver transições disparáveis no tempo atual, voltar ao passo 4;
7. avança o tempo para o menor tempo programado para que uma transição se torne habilitada para ser disparada;
8. analisa as transições sob a marcação atual;
9. as transições habilitadas são programadas para se tornarem disparáveis nos seus tempos correspondentes;
10. se não houver transições disparáveis, deve-se encerrar a simulação;
11. caso contrário, volta para o passo 2.

O simulador implementado analisa de redes de Petri Temporizadas, de modo que foi adotado um método de avanço do tempo direcionado a eventos. Para ilustrar esta funcionalidade, considere que em certo momento o simulador esteja analisando uma rede com três transições habilitadas (ou seja, existem três lugares com tempo de execução associado). Como o modelo considera que o tempo de execução está associado a um lugar, este é considerado o tempo de habilitação para o disparo da transição que sucede este lugar. Considere que destes trêslugareso menor tempo de execução é igual a 4.Utilizando-se o avanço de tempo orientado a eventos, o simulador verifica qual é o menor tempo para que qualquer uma das transições possa ser disparada e salta para aquele tempo.

O disparo de um passo, entretanto, é uma da tarefa complexa em uma RdP Temporizada, pois o simulador precisa resolver todos os conflitos da rede considerando os tempos limites relacionados aos lugares e transições envolvidos. Entretanto, este simulador trata de redes simplificadas, cujas informações foram obtidas a partir da WBS. Dessa forma, algoritmo utilizado para a resolução destes conflitos é descrito a seguir:

1. para cada lugar que habilita uma transição, deve-se checar se este lugar habilita mais de uma transição. Se verdadeiro, este lugar será adicionado na tabela de possíveis conflitos;
2. para cada lugar em possível conflito, calcula-se o somatório de todos os pesos dos arcos que partem deste lugar. Se o somatório dos pesos for igual ou inferior ao número de *tokens* presentes no lugar, o lugar não estará em conflito;
3. para cada lugar em conflito: enquanto ainda houver transições habilitadas, selecionar aleatoriamente uma das transições para disparar e garantir os seus *tokens*, recalculando a habilitação das restantes;
4. disparar todas as transições que tiveram seus *tokens* garantidos.

Conformese pode observar, os dois primeiros passos realizam a busca por conflitos. Os passos seguintes constituem a resolução destes conflitos.

6.4 Limitações do Modelo

Especificamente sobre o modelo desenvolvido, destacam-se as seguintes limitações:

- a proposta do modelo integrado SPIM se limita a projetos de desenvolvimento de software com metodologias baseadas no SPEM;
- as relações de precedência entre as tarefas somente podem ser expressas da forma fim-início. Esta decisão teve como objetivo simplificar o algoritmo CPM implementado;
- o modelo SPIM considera a integração com os conceitos propostos pelo modelo de reconfiguração dinâmica de projetos de software proposto por Callegari [CAL10];
- o modelo SPIM considera a adequação dos conceitos propostos pela integração do modelo SPIM considerando o uso da arquitetura multiagentes proposta por Schlösser [SCH10];
- Os fluxos organizacionais implementados na ferramenta SPIT foram desenvolvidos através do Windows Workflow Foundation. Integrações com outros mecanismos de workflow não estão previstas;
- A ferramenta SPIT foi desenvolvida considerando sua a integração com o software de gestão Microsoft Project 2013. Não foram consideradas integrações com outros softwares.

6.5 Considerações Finaisdo Capítulo

Este Capítulo apresentou o modelo computacional para a reconfiguração dinâmica de projetos de software, considerando a integração das atividades dos projetos com os fluxos organizacionais, proposto nesta tese de doutorado. Inicialmente, apresentou-se a estratégia para a reconfiguração dinâmica de projetos de software utilizada de maneira a ajustar o sequenciamento das atividades dos projetos, considerando um conjunto de restrições sobre as atividades, os recursos e as informações sobre as variáveis globais de projeto. Após a definição deste modelo computacional de reconfiguração dinâmica, foram apresentadas as funcionalidades desenvolvidas para o protótipo de ferramenta SPIT, que oferece suporte automatizado para o planejamento e replanejamento das atividades de projetos de software. O SPIT, atualmente, tem integrado ao seu ambiente os seguintes módulos: SPIM Validator, BackOffice, Workflow Integrator, Compiler e Simulator.

Após o desenvolvimento deste protótipo de ferramenta, foram desenvolvidos dois tipos de avaliações do modelo SPIM. A primeira avaliação foi realizada através de uma comparação, em um estudo experimental, entre a proposta de planejamento integrado

com relação à proposta tradicional de planejamento de projetos de software. A segunda avaliação foi realizada de forma analítica, ou seja, verificaram-se os resultados produzidos pelo modelo em função da ocorrência de eventos sobre cenários simulados.

7 AVALIAÇÃO DO MODELO

Para realizar a avaliação de modelos e produtos, onde o fator humano é considerado, a literatura fornece algumas abordagens com base em uma estratégia experimental. Pfleeger e Atlee [PFL09] sugerem as seguintes abordagens para avaliar processos, produtos e recursos:

a) análise de funcionalidades (*feature analysis*);
b) estudos de caso (*case studies*);
c) pesquisas (*surveys*); e
d) experimentos (*experiments*).

A análise de funcionalidades é usada para avaliar e classificar os atributos/características de um produto de software. No entanto, esta estratégia não avalia o comportamento dos dados em termos de causa e efeito. Estudo de caso é usado para organizar as informações sobre um caso/situação específico e, em seguida, analisar estas informações buscando padrões nestes dados e uma posterior análise, realizada através da comparação com outros casos/situações. *Survey* é um estudo retrospectivo para documentar as expectativas e resultados relacionados a determinadas situações. Esta técnica pode também ser aplicada para realizar uma avaliação empírica dos resultados através de indicadores qualitativos. No entanto, não há manipulação de variáveis de entrada neste estudo. Estudos experimentais, por outro lado, representam um tipo mais controlado de estudo, geralmente conduzido em laboratórios. Nesta abordagem, os valores das variáveis independentes (entradas do experimento) são manipulados para observar as alterações nos valores das variáveis dependentes (saídas do experimento). No final do experimento, os resultados são analisados, interpretados, apresentados e empacotados.

Nesta pesquisa, foram realizados dois tipos distintos de avaliações do modelo. O primeiro tipo de avaliação escolhido foi a utilização de um método formal de experimento. Este primeiro tipo de avaliação teve como propósito de verificar se o modelo SPIM atende as carências de integração identificadas no item “3.2 Necessidade de Integração”, ou seja: (1) Acesso às informações pertencentes aos outros departamentos da organização; (2)Identificação das relações de dependência entre as atividades dos fluxos

organizacionais e dos projetos de software; (3) Capacidade de minimizar distorções no planejamento de projetos (tais como, o aumento dos custos e atrasos nos prazos do projeto); e (4) Auxílio na identificação e mensuração dos custos indiretos do projeto. Desta forma, um primeiro estudo experimental usando o modelo SPIM foi conduzido por seis estudantes de graduação e quatro de pós-graduação de cursos da área da Ciência da Computação (os resultados foram publicados em [ROS12d]). Este primeiro experimento revela que a abordagem proposta pelo modelo SPIM fornece um mecanismo ao gestor para criar e realizar um plano de projeto mais preciso do que o método tradicional. No entanto, foram observadas algumas limitações nesta primeira experiência, tais como a utilização de estudantes de graduação. Desta forma, a pesquisa avançou para a realização de um segundo estudo experimental, agora envolvendo trinta e seis alunos de cursos de pós-graduação em Gerenciamento de Projetos (os resultados foram publicados em [ROS13]). Maiores detalhes sobre este segundo estudo experimental são apresentados nas seções a seguir. Cabe salientar, no entanto, que o experimento possui uma abordagem quantitativa [WOH00], de modo que um *Survey* também foi utilizado para avaliar os dados qualitativos. O segundo tipo de avaliaçãofoi realizado de forma analítica, ou seja, verificando-se os resultados produzidos pelo modelo em função da ocorrência de eventos sobre cenários simulados.

7.1 Avaliação através de Estudo Experimental

7.1.1 Planejamento e Execução do Estudo Experimental

Nesta pesquisa, as propostas de Juristo e Moreno [JUR03], Field [FIE05], e Wohlin et al. [WOH00] foram utilizados como guias para realizar esta experiência. Assim, o processo de experimentação adotado inclui as seguintes atividades principais:

(1) Definição dos objetivos dos experimentos;

(2) *Design* dos experimentos;

(3) Execução dos experimentos; e

(4) A análise dos resultados / dados coletados a partir dos experimentos.

Experimentos baseiam-se na análise de fenômenos.Assim, durante a definição dos objetivos, a hipótese geral é definida em termos de quais variáveis do fenômeno vão ser examinados. A atividade de *design* envolve a criação de uma espécie de plano segundo o qual o experimento deve ser realizado (vide ApêndiceD). Este plano decide quais variáveis devem ser examinadas, quais dados devem ser coletados, quantos

experimentos devem ser conduzidos e quantas vezes os experimentos devem ser repetidos. Na fase de execução, os experimentos são executados como indicado pelo modelo de design selecionado. Uma vez que os experimentos foram realizados, os dados coletados durante os experimentos devem ser analisados. Esta análise visa encontrar relações entre os resultados do estudo. Cada atividade principal do experimento é apresentada nas seções seguintes.

7.1.1.1 Definição dos Objetivos do Experimento

Esta atividade corresponde à definição das hipóteses e objetivos a serem alcançados quanto à solução do problema. A técnica *Goal-Question-Metric (GQM)* [SOL99] foi utilizada neste estudo, estabelecendo o objetivo geral e a forma de medição dos dados. Seguindo esta abordagem, foi decidido que o objetivo desta pesquisa é comparar, conforme o RUP, a precisão e o esforço usando o modelo de planejamento integrado SPIM em comparação com o modelo tradicional de planejamento de projeto de software.

Uma vez definido o objetivo principal, o passo seguinte é definir as métricas básicas que podem representar o valor das entradas e os resultados deste experimento. De acordo com a técnica GQM, duas questões foram identificadas:

(1) O esforço para realizar o planejamento das atividades do projeto de software usando o modelo integrado SPIM é igual ao esforço para realizar o planejamento das atividades de acordo com o modelo tradicional?

(2) A precisão no planejamento do cronograma de projetos de software - relacionado à atribuição de prazos e recursos, considerando a integração do projeto com os fluxos organizacionais através do modelo integrado SPIM – é igual à precisão para realizar o planejamento de acordo com o modelo tradicional?

A métrica associada à primeira pergunta corresponde ao esforço medido pela razão entre o tempo gasto em minutos por cada participante durante o planeamento das atividades de software em cada abordagem. A métrica associada à segunda questão corresponde à precisão relacionada à atribuição de prazos e recursos sobre a programação das atividades do projeto utilizando cada uma das abordagens, evitando assim a ocorrência de certos tipos de riscos no projeto. Para este estudo, a precisão foi

definida como a razão entre a pontuação feita pelos participantes e da pontuação total possível, de acordo com um modelo previamente desenvolvido pelos pesquisadores.

7.1.1.2 Design do Experimento

A atividade de *design* é necessária para formalizar as hipóteses, determinar as variáveis independentes e dependentes, selecionar dos participantes, fazer a preparação do experimento e validar o experimento.

Para realizar o experimento sobre o planejamento de projetos de software, foi escolhido um contexto que envolve estudantes de duas instituições de ensino superior distintas, a saber: oInstituto Brasileiro de Gestão de Negócios (IBGEN) e a Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Assim, este estudo envolveu trinta e seis alunos dos cursos de pós-graduação em Gerenciamento de Projetos. Foi utilizada uma amostra não probabilística para a seleção dos indivíduos, de modo que foram escolhidas para o experimento as pessoas mais convenientes (*convenience sampling*) e também indivíduos de populações diferentes, nesse caso, foram escolhidos alunos de pós-graduação de duas instituições de ensino superior (*quote sampling*).

Este experimento consiste em uma abordagem "*in-vitro*" e "*off-line*" em que os participantes realizaram o experimento em um ambiente controlado e fora do contexto real de projetos de desenvolvimento de software. Esta abordagem pode reduzir os riscos e os custos não abrangidos no âmbito desta pesquisa neste momento. Além disso, com base na definição informal prévia das duas questões desta pesquisa, foi possível formalizar as duas hipóteses e definir suas respectivas medidas de avaliação.

A primeira hipótese está relacionada com o esforço dos gestores no planejamento das atividades e recursos para projetos de software. Então, a primeira hipótese nula (H_0) é a seguinte: o esforço envolvido no planejamento das atividades do projeto de software usando o modelo integrado SPIM é igual ao esforço de fazer o planejamento de atividades de acordo com o modelo tradicional. O esforço será medido pelo tempo gasto em minutos com o planejamento das atividades para projetos de desenvolvimento de software em cada abordagem, isto é, a diferença entre o tempo de início e o tempo final de cada abordagem, onde:

- Δ_{tspim}: representa a variação do tempo gasto em minutos para o planejamento das atividades do projeto usando o modelo SPIM;

- Δ_{ttrad}: representa a variação do tempo gasto em minutos para o planejamento das atividades do projeto usando o modelo tradicional;

A fórmula para o cálculo esforço é a seguinte: H_0: $\Delta_{tspim} = \Delta_{ttrad}$. Como hipótese alternativa (H_1), o esforço envolvido com o planejamento das atividades do projeto usando o modelo SPIM não é igual ao esforço envolvido na utilização do modelo tradicional. Isto é, H_1: $\Delta_{tspim} \neq \Delta_{ttrad}$.

A segunda hipótese desta pesquisa está relacionada com a precisão dos gestores em planejar as atividades e recursos de projetos de software. Então, a segunda hipótese nula (H_0) é a seguinte: a precisão no planeamento do cronograma de projetos de software em relação à atribuição de prazos e recursos, considerando a integração dos projetos com os fluxos organizacionais, através do modelo integrado SPIM é igual à precisão em realizar o planeamento no modelo tradicional. A precisão será avaliada pela relação de pontuação dos participantes com a pontuação total possível, onde:

- P_{spim}: Precisão associado com o planejamento usando o modelo SPIM;
- P_{trad}: Precisão associado com o planejamento tradicional;

A fórmula para a precisão do cálculo é a seguinte: H_0: $P_{tspim} = P_{ttrad}$. Como hipótese alternativa (H_1), a precisão no planejamento das atividades do projeto usando o modelo SPIM não é igual à precisão realizando o planejamento de acordo com o modelo tradicional. Isto é, H_1: $P_{tspim} \neq P_{ttrad}$.

Depois de estabelecer as hipóteses, foram identificadas algumas características importantes para este experimento. Consequentemente, é fundamental estar familiarizado com a terminologia utilizada em experimentos de software. Uma unidade experimental é a porção do material experimental em que um tratamento é aplicado. Em outras palavras, as unidades experimentais são os objetos sobre os quais o experimento é executado. Neste caso, cinco projetos de desenvolvimento de software são as unidades experimentais (ou objetos experimentais) desta pesquisa. Estes cenários são apresentados mais adiante no texto.

A pessoa que aplica as técnicas de experimentação nas unidades experimentais é chamada de sujeito experimental. O resultado de um experimento de engenharia de software pode variar muito dependendo de quem são os sujeitos, em termos de sua experiência na aplicação de algumas técnicas, e até mesmo sobre o estado emocional do sujeito no momento de executar o experimento. Portanto, eles têm que ser cuidadosamente considerados durante o *design* do experimento. Nesta pesquisa, foi

definido que os sujeitos experimentais seriam estudantes de pós-graduação com experiência anterior em projetos de desenvolvimento de software.

O resultado de um experimento é chamado de variável de resposta ou variável dependente. Esta variável deve ser quantitativa em experimentos realizados em laboratório. As variáveis de resposta deste experimento estão relacionadas com o esforço e a precisão no planejamento das atividades de projetos de software. Cada valor da variável de resposta reunido em um experimento é chamado de observação, e a análise de todas as observações vai decidir se as hipóteses a serem testadas podem ser validadas ou não.

Além disso, é necessária a definição de quaisquer características (chamado de parâmetros) que não são desejadas para influenciar o resultado do experimento ou da variável de resposta. Neste caso, a complexidade das tarefas propostas e da ordem de execução dos projetos deve permanecer inalterada a entre um experimento para o outro. Por outro lado, as características do projeto que variam intencionalmente durante o estudo experimental e que afetam o resultado do experimento são chamadas de fatores ou variáveis independentes. Cada fator tem várias alternativas possíveis. Neste experimento, há um fator a ser analisado (métodos de planejamento de projetos) e duas alternativas: o método tradicional de planejamento do projeto e o método usando o modelo integrado de planejamento SPIM. Portanto, o experimento visa examinar a influência destas alternativas sobre o valor da variável de resposta. Algumas características as quais se gostariam que permanecessem invariáveis (parâmetros), no entanto, podem variar durante o experimento. Essas variações indesejáveis são chamadas de variáveis de bloqueio. Neste experimento, o nível de experiência em planejamento de projetos é uma variável de bloqueio.

Depois de estabelecer os parâmetros, fatores, variáveis de bloqueio e variáveis de resposta, um tipo de *design* de experimento teve que ser escolhido. Existem diferentes *designs* de experimentos em função do objetivo do estudo experimental, do número de fatores, das alternativas desses fatores e do número de variações indesejáveis, entre outras características. Para este experimento, conforme mencionado anteriormente, cinco diferentes cenários de projeto de desenvolvimento de software foram desenvolvidos com o objetivo de abordar os riscos do projeto de software diferentes (vide Apêndice C). O primeiro cenário diz respeito à compatibilidade da atribuição do papel das partes interessadas envolvidas com o tipo de atividade (de gestão ou de produção). O segundo

cenário está relacionado com a interação entre os fluxos organizacionais para a aquisição de novo hardware ou software durante o projeto. O terceiro cenário está relacionado com o risco de identificar que o pessoal mais qualificado não está disponível em momentos críticos. O quarto cenário está relacionado com o risco de identificar que a formação necessária para o pessoal não está disponível. O quinto cenário está relacionado com o risco de se identificar que os componentes de software que devem ser reutilizados contêm defeitos, limitando a sua funcionalidade. A validação externa destes cenários foi realizada através da realização de um pré-teste do experimento (vide Apêndice D) com um profissional experiente da área de gestão de projetos.

Considerando todas as características desta pesquisa, foi necessário descobrir qual das duas alternativas do fator (tradicional ou SPIM) apresentou melhores resultados em relação a uma dada variável de resposta (esforço e precisão). Em seguida, foi escolhido o *design* chamado de um fator (*one-factor design*). Este tipo de *design* consiste em comparar a variável de resposta para cada alternativa de um dado número de unidades experimentais. Aplicando as duas alternativas para a mesma unidade experimental significa que cada alternativa deve ser aplicada para o mesmo projeto em questão. Se cada alternativa é utilizada no mesmo projeto, duas equipes semelhantes são necessárias. Além disso, a atribuição das alternativas aos experimentos deve ser randômica, a fim de assegurar a validade das alternativas. Portanto, algum tipo de seleção aleatória deve ser usado, por exemplo, jogando um dado, retirando cartas de uma caixa, etc. Neste caso, o experimentador colocou trinta e seis cartas (metade preta e metade vermelha) em uma caixa; as cartas vermelhas corresponderiam à utilização do método tradicional de planeamento de projeto e as pretas ao uso do método SPIM. O experimentador embaralhou as cartas e permitiu que cada sujeito retirasse uma carta para cada unidade experimental (projeto de desenvolvimento de software). Também foi utilizado o princípio de equilíbrio (*balancing principle*), de modo que cada proposta de planeamento de projetos de software foi realizada pelo mesmo número de participantes (dezoito participantes para cada proposta).

Ao projetar experimentos de engenharia de software, algumas considerações específicas devem ser levadas em consideração, especialmente aquelas relativas às competências/habilidades, também às circunstâncias psicológicas e outras características dos indivíduos que participam do experimento. O *design* de um fator (*one-factor*) possui uma limitação, uma vez que considera o fato de que o mesmo sujeito aplica o mesmo

método mais de uma vez, o que pode levar os participantes a se tornarem mais familiarizados com o método (o sujeito pode aprender a aplicar o método ao longo do tempo). Outra consideração para projetos experimentais é chamada de efeito tédio (*boredom effect*), que ocorre quando os indivíduos ficam entediados ou cansados com o experimento ao longo do tempo e colocam menos esforço para a execução do experimento com o passar do tempo. Por esta razão, este experimento foi realizado durante o tempo de aula dos alunos. Finalmente, foi considerado o caso oposto ao efeito tédio: o efeito entusiasmo (*enthusiasm effect*). Este ponto de motivação pode surgir quando um novo método está para ser testado contra um método tradicional. Os indivíduos que aplicam o método tradicional podem não estar motivados para fazer um bom trabalho, enquanto que aqueles que aplicam o novo método podem estar mais inspirados e motivados em aprender algo novo. Para evitar essa situação, foi realizado um experimento às escuras (*blind experiment*), uma tática em que os sujeitos não estão familiarizados tanto com as hipóteses formuladas ou objetivos do experimento.

7.1.1.3 Execução do Experimento

Esta terceira atividade envolve a preparação, implementação e validação dos dados obtidos no experimento. A realização do experimento ocorreu em novembro de 2011, quando um conjunto de participantes realizou o experimento em um ambiente controlado (laboratório de computadores dasinstituições de ensino superior). O problema estudado correspondeu a cinco cenários que simularam situações ocorridas em projetos de desenvolvimento de software (*toy example*).

Inicialmente, todos os participantes receberam um breve treinamento sobre o modelo de planejamento integrado SPIM e, posteriormente, tiveram a oportunidade de testar as principais características do módulo SPIM Validator em um projeto de exemplo. Mais tarde, eles tiveram a oportunidade de fazer as primeiras perguntas sobre o trabalho proposto. O experimento começou somente depois que se confirmou que todos os participantes entenderam como executar o SPIM. Em seguida, eles foram apresentados à mesma descrição de cada cenário e foram convidados a realizar o planejamento do projeto correspondente - alguns usando o método tradicional e outros com a ferramenta SPIT. A fim de evitar possíveis distorções nos resultados obtidos, tanto na avaliação com o SPIT quanto na fase de resolução do questionário, não ocorreu qualquer interação com o entrevistador.

Inicialmente, todos os participantes receberam um e-mail convidando-os a participar deste estudo experimental. Neste convite foi explicado que o evento contaria com a apresentação do modelo SPIM e a realização de uma atividade prática onde os participantes teriam a oportunidade de realizar exercícios com base em situações típicas de gerenciamento de projetos. Reforçou-se, também, que esta atividade prática era caracterizada como um estudo experimental, cujos resultados permitiriam aos pesquisadores realizar uma análise das dificuldades impostas aos gerentes de projeto na solução dos problemas propostos com e sem o uso do modelo SPIM. Este foi um evento gratuito e com lugares limitados (máximo de 40 participantes). Consequentemente, o experimento envolveu apenas os alunos de pós-graduação que tinham algum interesse na área de gerenciamento de projetos. Para participar neste evento os convidados tiveram que acessar o link para o evento e criar uma conta de acesso. Assim, um site foi desenvolvido para armazenar os questionários deste experimento (Figura 51) com o objetivo de manter a integridade dos dados obtidos durante sua execução (vide ApêndiceC). Ainda assim, um sistema de controle de acesso assegurado que cada participante tinha acesso apenas às questões que foram designados por eles, diminuindo as chances de interferência em suas respostas.

Figura 51. Site sobre o experimento do modelo SPIM

7.1.1.4 Análise do Experimento

Esta subseção descreve como os dados coletados do experimento foram examinados, a fim de auxiliar nas conclusões desta pesquisa. Existem diferentes técnicas de análise, dependendo das características dos dados recolhidos e sobre o

*design*aplicado. Os métodos de análise podem ser divididos em dois blocos principais: métodos paramétricos e não paramétricos. Testes paramétricos requerem uma suposição paramétrica, tal como a normalidade, e testes não paramétricos são frequentemente utilizados no lugar dos seus homólogos paramétricos quando certas suposições sobre a população subjacente não são atendidas.

A escolha de cada técnica de análise a ser aplicada sobre o experimento depende do tipo de escala da variável de resposta. Assim, quando a variável de resposta é nominal ou ordinal, os métodos que devem ser utilizados durante a análise correspondem aos do grupo dos não paramétricos e quando a variável de resposta é medida em uma escala de intervalo ou razão, métodos paramétricos e não paramétricos devem ser aplicados. No entanto, os testes paramétricos são estatisticamente mais poderosos do que os métodos não paramétricos [MIL94]. Assim, os testes paramétricos devem ser aplicados, idealmente, para analisar os dados coletados a partir dos experimentos. Porém, se estes testes paramétricos não são conclusivos, então a análise vai ter que recorrer à aplicação de testes não paramétricos, embora sejam menos poderosos estatisticamente. Considerando estes dois tipos de técnicas de análise, a elaboração das conclusões foi elaborada através da rejeição das hipóteses nulas com testes paramétricos e/ou através de sua aceitação com os testes não paramétricos.

Para o teste das hipóteses, em um contexto de um fator e dois tratamentos, a literatura sugere que o teste de significância chamado teste T para duas amostras independentes (se for realizado um teste paramétrico) ou o teste *Mann-Whitney* (se for um teste não paramétrico). Esta definição deve ser tomada depois da verificação se a distribuição é normal ou não (pelo teste *Shapiro-Wilk*) e da verificação da variação dos dados obtidos através da execução do experimento (teste *Levene*).

Esta fase também envolve documentar os resultados de modo a permitir a sua replicação em diferentes circunstâncias. O desempenho do experimento em diferentes contextos permite a aquisição de novos conhecimentos sobre os conceitos estudados. Na seção seguinte, será detalhada a análise dos dados obtidos neste experimento.

7.1.2 Análise dos Resultados do Experimento

A primeira análise dos dados brutos obtidos no experimento está relacionada com os seus tipos de escala. Ao trabalhar com variáveis e métricas,precisa-se considerar os diferentes tipos de escala de medição. Os tipos de escala mais comuns são: nominal,

intervalo, ordinal e razão [FEN97], [KIT96]. O tipo de escala determina o tipo de método de análise dos dados que deve ser aplicado para se obter as respectivas conclusões sobre o experimento. Assim, a variável independente em relação aos métodos de planeamento de software é representada como sendo uma escala nominal e as variáveis dependentes sobre a precisão são representadas por uma escala de razão.

As variáveis dependentes são caracterizadas pela escala de razão, o que permite o cálculo da normalidade e homocedasticidade, necessários para definir o tipo de teste das hipóteses (paramétrico ou não paramétrico). Uma distribuição normal é uma distribuição simétrica, geralmente representada graficamente por uma curva em forma de sino, que atinge o pico em torno do seu centro. Homocedasticidade refere-se à suposição de que a variável dependente exibe quantidades semelhantes de variância em toda a gama de valores para uma variável independente.

Inicialmente, foi realizada a verificação de cada hipótese nula. A hipótese nula (H_0) está relacionada com a aleatoriedade dos resultados observados, isto é, se isso for verdade, estatisticamente, os resultados do experimento evidenciam para serem ocasionais (nenhuma conclusão pode ser feita). A hipótese alternativa (H_1) é aquela que vai ser aceita se a hipótese nula for rejeitada. No entanto, deve-se notar que o nível de significância do teste foi fixado em 5%. Logo, o menor nível de significância que pode rejeitar a hipótese nula (*p-value* ou α) é de 0,05. As análises apresentadas neste experimento foram realizadas com o software *Statistical Package for Social Sciences* (SPSS) [IBM11].

7.1.2.1 Primeira Hipótese: Esforço

Através de uma análise inicial da distribuição, pode-se avaliar o comportamento das amostras. Inicialmente, o conjunto de dados foi analisado para ver se alguma informação (observação) estava demasiadamente longe do restante. As observações que se distanciaram do resto são normalmente chamadas de *outliers*. Um *outlier* é aquela observação que parece se desviar acentuadamente dos outros membros da amostra em que ela ocorre [GRU69]. A identificação destes valores extremos foi obtida pelo gráfico *boxplot* (ver Figura 52). Através gráfico *boxplot* foi possível observar a maneira como as variáveis estão distribuídas em relação à homogeneidade dos dados, os valores de tendência central, os valores máximos e mínimos e os outliers. Assim, pode-se identificar que os valores relacionados ao esforço (em minutos) para planejar os projetos do

experimento usando o modelo SPIM são mais heterogêneos do que aqueles que utilizaram o modelo tradicional de planejamento. Ainda, pode-se observar a ausência de *outliers*.

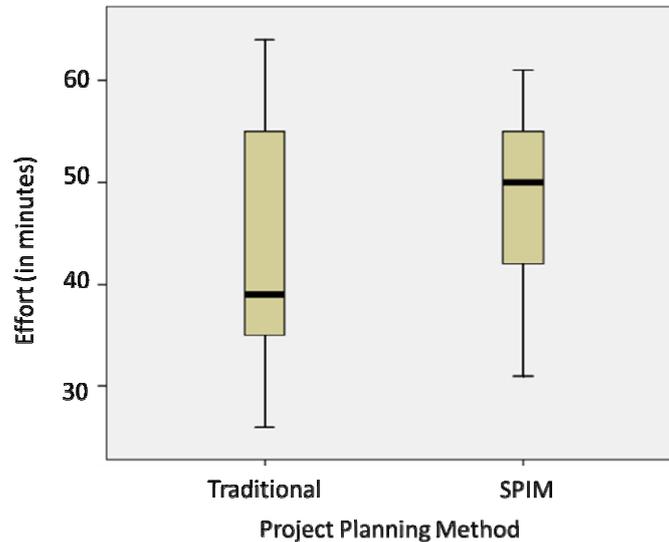



Figura 52. Gráfico *boxplot* sobre o esforço necessário para o planejamento dos projetos

Após esta primeira análise, deve-se verificar se os dados têm ou não uma distribuição normal. Para atingir este objetivo, uma hipótese nula e uma hipótese alternativa foram definidas, como se segue: H_0: é uma distribuição normal; H_1: não é uma distribuição normal. O teste de *Shapiro-Wilk* foi usado para verificar se a distribuição dos dados é normal (a Tabela 9 apresenta os resultados deste teste).

Tabela 9: Resultados do teste de *Shapiro-Wilk* para a variável esforço

Variável	**Método**	**Estatística**	**Grau de Liberdade**	**Significância**
Esforço	Tradicional	0.923	18	0.143
	SPIM	0.947	18	0.381

Com base na Tabela 9, pode-se observar que o nível de significância em ambas as amostras (tradicional e SPIM) são maiores do que o nível de significância que pode rejeitar a hipótese nula (0,05 ou 5%). Com esta informação, não há nenhuma evidência para rejeitar a hipótese nula, de maneira que a distribuição é normal. Este resultado é confirmado pela análise do histograma e do gráfico de probabilidade normal (Figura 53), onde os histogramas mostram uma distribuição simétrica quando se utiliza tanto o método tradicional de planejamento quanto usando o SPIM.

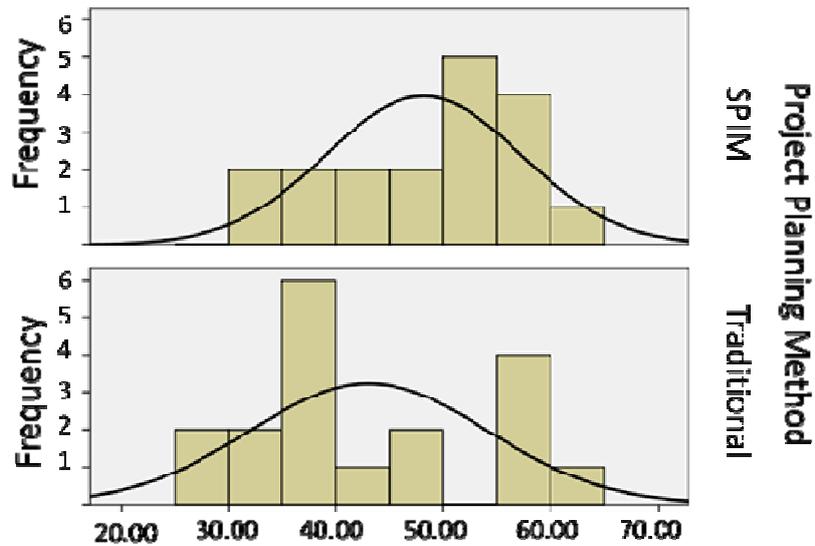



Figura 53. Histograma sobre o esforço para fazer o planejamento de atividades utilizando os dois métodos

O teste T também assume que a variabilidade de cada grupo é aproximadamente igual. Desta forma, foi necessário analisar a variância das duas amostras. Com este objetivo, duas hipóteses foram definidas: H_0: As variâncias são iguais; H_1: As variâncias não são iguais. O teste de *Levene* para igualdade de variâncias (ver Tabela 10) mostra se os critérios necessários para o teste T foram cumpridos.

Tabela 10: Teste de *Levene* de igualdade de variâncias para a variável esforço

Variável	Critério	Significância
Esforço	Variâncias iguais	0.271
	Variâncias diferentes	0.271

De acordo com os resultados da Tabela 10, a significância (*p-value*) do teste de *Levene* é 0,271. Se este valor for igual ou inferior ao nível de significância (α) para o teste (neste caso 0,05), então a hipótese nula, cuja variabilidade dos dois grupos é igual, pode ser rejeitada, o que implica que as variâncias são diferentes. Se o *p-value* é maior do que o nível α, então, variâncias iguais são assumidas. Neste caso, 0,271 é maior do que α, de modo que se assumiu que as variâncias são iguais.

Uma vez que se identificou que a distribuição era normal e as variâncias eram iguais, foi aplicado o teste T (ver resultados da Tabela 11). O teste T para amostras independentes pode ser usado para identificar se duas médias são diferentes uma do outra quando as duas amostras, as quais as médias foram calculadas, são compostas de diferentes indivíduos.

Tabela 11: Teste T para a variável esforço

Variável	**Critério**	**T**	**Grau de Liberdade**	**Significância (2 tailed)**
Esforço	Variâncias iguais	-1.511	34	0.140
	Variâncias diferentes	-1.511	32.699	0.140

Este é um teste chamado *two-sided test*, em que o valor de p=0,140 é diretamente comparado com α = 0,05 (nível de significância). Uma vez que *p-value* = 0,140> 0,05, H_0 não é rejeitada. Assim, não há nenhuma evidência estatística para rejeição da hipótese de que a esforço médio para fazer o planejamento das atividades usando o modelo tradicional é igual ao esforço realizado com o uso do modelo de SPIM. Observou-se que a média de esforço, em minutos, para realizar a programação das atividades usando o modelo de SPIM foi de cerca de 48 minutos enquanto que o uso do modelo tradicional que foi de cerca de 43 minutos (a diferença entre estas duas medidas - diferença média - é de somente 5 minutos). Desta forma, o pequeno aumento no esforço, ocasionado pelo preenchimento dos campos adicionais propostos pelo modelo SPIM, é compensado pela quantidade de informações extras que podem auxiliar o gestor a aumentar a precisão do planejamento dos projetos (esta verificação pode ser observada pela análise da segunda hipótese – precisão).

Durante a definição dos objetivos deste experimento (item 7.1.1.1 - Definição dos Objetivos do Experimento), identificou-se como um dos objetivos verificar se o esforço para realizar o planejamento das atividades do projeto de software usando o modelo integrado SPIM é igual ao esforço para realizar o planejamento das atividades de acordo com o modelo tradicional. Conforme os resultados apresentados, pode-se concluir que, estatisticamente, não há diferença significativa em relação ao esforço para se realizar o planeamento de projetos de software utilizando o método tradicional em relação ao método SPIM.

7.1.2.2 Segunda Hipótese: Precisão

O gráfico *boxplot* foi utilizado para identificar *outliers* (ver Figura 54) de maneira similar à análise da primeira hipótese. De acordo com este gráfico observou-se que a variável de precisão não tem *outliers*.

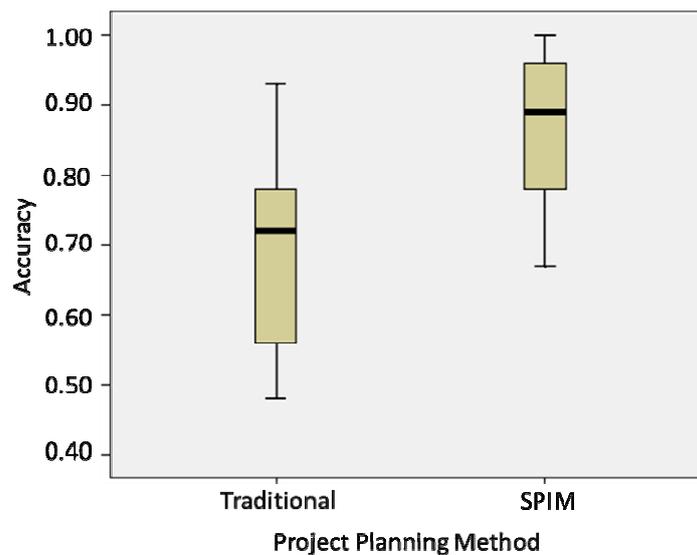



Figura 54. Gráfico *boxplot* em relação à precisão para o planejamento dos projetos

O passo seguinte foi identificar se os dados tem uma distribuição normal. Para atingir este objetivo, as hipóteses seguintes foram definidas: H_0: é uma distribuição normal; H_1: não é uma distribuição normal. O teste de *Shapiro-Wilk* mostra se o critério para o teste T está sendo cumprido (ver Tabela 12).

Tabela 12: Resultados do teste de *Shapiro-Wilk* para a variável precisão

Variável	**Método**	**Estatística**	**Grau de Liberdade**	**Significância**
Precisão	Tradicional	0.926	18	0.195
	SPIM	0.912	18	0.022

Considerando os mesmos pressupostos para a variável esforço, comparando diretamente *p-value* com α (nível de significância), observa-se que *p-value*= 0,022> 0,05 faz rejeitar H_0 quando se considera a utilização do modelo SPIM. Portanto, um teste paramétrico (como o teste T) não poderia ser usado. No entanto, o teste de *Mann-Whitney* pode ser usado nestes casos. Ele é teste não paramétrico de amostras independentes análogo ao teste T e pode ser usado quando não se assume que a variável dependente possui uma distribuição normal (somente se assume que a variável é, pelo menos, ordinal). Assim, o teste de *Mann-Whitney* para duas amostras independentes é usado para verificar se as diferenças observadas entre as médias em dois grupos independentes são estatisticamente significativas. Para atingir este objetivo, as seguintes hipóteses foram definidas: H_0: Não há diferença entre a média das duas amostras; H_1: Há uma diferença entre a média das duas amostras. Os resultados do teste de *Mann-Whitney* podem ser vistos na Tabela 13.

Tabela 13: Teste não paramétrico de *Mann-Whitney* para a variável de precisão

Variável	Mann-Whitney U	Wilcoxon W	Z	Significância (2-tailed)	Significância [2*(1-tailed Sig.)]
Precisão	66.500	237.500	-3.046	0.002	0.002a

Uma vez que o grau de significância (0,002) é menor do que o nível de significância dado (0,05), a hipótese H_0 foi rejeitada. Com base nos resultados apresentados para a variável precisão, entende-se que existe uma diferença entre o esforço médio para realizar o planeamento entre os métodos tradicional e SPIM. No entanto, com base nos resultados do teste de *Mann-Whitney* apenas a hipótese nula pode ser rejeitada, não foi possível avaliar as hipóteses alternativas. Alternativamente, a análise descritiva das médias das amostras é comparada como mostrado na Tabela 14.

Tabela 14: As estatísticas descritivas para a variável precisão

Método	Média
Tradicional	0.7725
SPIM	1.5000

Cabe relembrar o segundoobjetivo deste experimento, listado no item (7.1.1.1 - Definição dos Objetivos do Experimento), que trata de verificar se a precisão no planejamento do cronograma de projetos de software - relacionado à atribuição de prazos e recursos, considerando a integração do projeto com os fluxos organizacionais através do modelo integrado SPIM – é igual à precisão para realizar o planejamento de acordo com o modelo tradicional. Desta forma, comparando-se os valores médios das duas abordagens, pode-se concluir que a precisão para realizar o planeamento usando o modelo SPIM foi maior do que usando o modelo tradicional.

7.1.2.3 Análise Qualitativa

A fim de avaliar os conceitos advindos do modelo integrado SPIM sobre a sua aceitação e aplicabilidade, também foi realizada uma avaliação qualitativa exploratória. No final da execução do experimento, cada participante respondeu a um questionário (vide Apêndice E), produzido em conformidade com Rea e Parker [REA05]. Este questionário possui 17 questões, onde as 8 primeiras estão focadas em mapear o conhecimento individual de cada gerente, o restante das questões foi utilizado para coletar contribuições e sugestões dos gerentes considerando o modelo SPIM no processo de planejamento de projetos de software.

Uma análise dos resultados obtidos a partir das questões relacionadas com o perfil dos indivíduos mostra que 52,77% deles possuem uma experiência em gestão de projetos entre dois e cinco anos e 22,22% possuem experiência entre cinco e dez anos. Além disso, 36,11% da amostra qualificaram-se como tendo pouca experiência em gerenciar de projetos, enquanto que o restante, 63,89%, qualificou-se como tendo conhecimentos entre moderado e avançado. Além disso, 72,22% dos indivíduos classificaram o seu conhecimento com relação aos processos de desenvolvimento de software como sendo moderado ou avançado. Isso indica um nível suficiente de experiência destes sujeitos relacionado à gestão de projetos.

A análise do modelo SPIM iniciou com a avaliação dos participantes sobre os benefícios diretos em realizar o planejamento integrado de atividades gerenciais e produtivas em projetos de software. Os resultados são mostrados na Tabela 15.

Tabela 15: Os benefícios percebidos na realização do planejamento integrado de atividades gerenciais e produtivas

Percepção	Resp.	%
Redução do tempo durante o processo de elaboração do projeto	19	52.77%
Identificação das dependências entre as atividades gerenciais de apoio e as atividades produtivas	36	100%
Identificação e mensuração dos custos indiretos do projeto devido às atividades gerenciais de apoio	22	61,11%
Capacidade de acessar informações dos fluxos de trabalho empresa	27	75%
A capacidade de minimizar distorções durante o planejamento através das atividades gerenciais de apoio	36	100%
Permite que os gerentes de projeto antecipem as necessidades decorrentes das áreas de suporte da organização durante o planejamento do projeto	32	88.80%
Faz a distinção explícita entre as atividades de um projeto de software e as atividades que pertencem a outros departamentos dentro da organização	36	100%

De acordo com a segunda e a última linha da Tabela 15, todos os participantes informaram que o planejamento integrado permite a identificação das dependências ocultas entre as atividades gerenciais de apoio e as atividades produtivas, evitando frequentes distorções no planejamento dos projetos. A visibilidade da integração das atividades gerenciais de apoio com as atividades do projeto de software (seja produtiva ou gerencial) também foi identificada como um forte benefício do SPIM por todos os entrevistados.

A obscuridade em identificar as atividades gerenciais de apoio durante o planejamento do projeto pode afetar negativamente o projeto. Quando questionados se concordavam ou não sobre a natureza distinta dos três tipos de atividades, a maioria dos entrevistados (75%) respondeu que o modelo SPIM ajuda os gestores a acessar as informações dos fluxos organizacionais. Além disso, 88,80% dos entrevistados concordaram que o modelo SPIM contribui na identificação das dependências das atividades entre o fluxo de trabalho do projeto e o fluxo organizacional, permitindo a previsão das necessidades que surgem nas áreas de apoio da organização durante o planejamento do projeto.

Como consideração final, a maioria dos entrevistados concluiu que o modelo SPIM contribui na identificação e mensuração dos custos indiretos do projeto, devido ao conceitorelacionado às atividades gerenciais de apoio. Além disso, quase 50% dos entrevistados notou uma redução no tempo durante o processo de elaboração do projeto. Os resultados coletados na pesquisa reafirmam os benefícios que o modelo SPIM fornece na resolução dos problemas relacionados com a definição inadequada de tarefas devido à obscuridade em visualizar a interdependência entre os fluxos organizacionais e os fluxos dos projetos.

7.2 Avaliação Analítica através de Cenários

Esta segunda avaliação foi realizada de forma analítica, ou seja, verificando-se os resultados produzidos pelo modelo em função da ocorrência de eventos sobre cenários simulados. O objetivo desta avaliação foi de mostrar a integração do modelo SPIM com o modelo de referência para reconfiguração dinâmica de projetos de software proposto em Callegari [CAL10]. Este modelo de referência sugere tratar as alterações ocorridas durante a execução dos projetos de software através de um ciclo de chamada de propostas (*Call For Proposal* – CFP). Resumidamente, a CFP é composta pelas posições detrabalhos comprometidas em função de um determinado evento.Através desta informação, é possível identificar os recursoscandidatos para estas posições de trabalho.Ainda, em momento posterior à composição da CFP, esta proposta é enviadaaos *solvers* presentes no modelo para o envio de propostas de realocação.

Neste momento, observa-se a necessidade de esclarecer como os cenários foram detalhados e quais são as convenções adotadas para embasá-los. Cabe salientar que tais convenções são suportadas pelo modelo de reconfiguração dinâmica de projetos de

software [CAL10]. Neste modelo, cada projeto é representado por um diagrama de rede através do Método do Diagrama de Precedência (MDP). O MDP é um método usado no Método do Caminho Crítico (CPM) para a construção de um diagrama de rede do cronograma do projeto [PMI08]. Esse método utiliza caixas ou retângulos (também chamados de nós ou blocos) para representar as tarefas e as conecta por setas para ilustrar as dependências. O MDP inclui quatro tipos de relações de dependências [PMI08]:

- Término para início: indica que o início de uma tarefa sucessora depende do término de sua predecessora. Esse é um dos tipos de dependência mais utilizado para representar as relações no MDP.
- Término para término: indica que o término de uma tarefa sucessora depende do término de sua predecessora.
- Início para início: indica que o início de uma tarefa sucessora depende do início de sua tarefa predecessora.
- Início para término: indica que o término de uma tarefa sucessora depende do início de sua tarefa predecessora.

Embora o MPD apresente estes quatro tipos de relações de dependências, para os propósitos especificados para o modelo SPIM, utiliza-se apenas a relação término para início. Os demais tipos de dependência não foram incorporados para simplificar o algoritmo CPM implementado. O PMBOK Guide, entretanto, indica que normalmente são utilizadas relações de dependências término para início em projetos desta natureza. Ainda, para melhor representar a inter-relação dos projetos de software com as atividades pertencentes aos fluxos organizacionais, optou-se descrever os cenários utilizando as redes de Petri. Conforme salientado anteriormente, a rede de Petri é uma ferramenta gráfica e matemática para a modelagem, análise e verificação de sistemas. Assim,a principal diferença entre as redes de Petri e as outras ferramentas de modelagem é a presença de fichas (*tokens*), os quais são utilizados para simular as atividades em um sistema dinâmico.

Outra informação importante consiste no cálculo dos intervalos de início e término das tarefas componentes de um projeto, o qual é realizado através do Método do Caminho Crítico (*Critical Path Method* – CPM) [UHE03]. Esses intervalos auxiliam na identificação do conjunto de tarefas que não podem ser adiadas em função do prazo do projeto. A esse conjunto de tarefas críticas denomina-se “caminho crítico”, o qual constitui o caminho mais longo do projeto [UHE03]. As demais tarefas, entretanto, possuem folga e

dispõem de maior flexibilidade no seu agendamento função do intervalo proporcionado pela folga. Segundo Prado [PRA04], é importante observar o caminho crítico, pois as tarefas que o compõem devem ser cumpridas no prazo estabelecido, de forma que não haja atrasos no prazo do projeto.

As unidades de tempo serão representadas em unidades discretas. Na Figura 55pode ser visto um exemplo gráfico do diagrama de rede Petri que será utilizado para a descrição dos projetos de software e dos fluxos organizacionais.

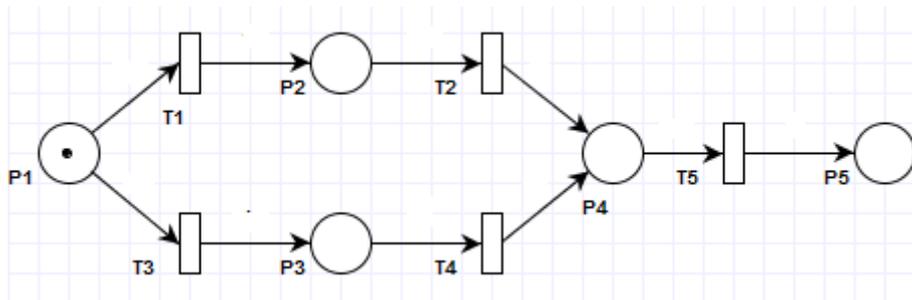



Figura 55. Exemplo de um diagrama de rede de Petri

Considera-se nesta pesquisa a execução de vários projetos simultaneamente, de modo que, durante a descrição dos cenários, assume-se que os projetos estão sendo executados de forma paralela e sem dependência entre eles.

7.2.1 Descrição dos Cenários

A elaboração de cada cenário foi realizada da seguinte forma: serão apresentadas configurações de projetos e eventos hipotéticos que ocorrerão sobre estes projetos. Assim, uma CFP (*Call For Proposal*) será gerada para cada evento hipotético que ocorrer. O processo de geração das CFP, entretanto, não é abordado nesta pesquisa.Ressalta-se, dessa forma, que não é escopo desta pesquisa apresentar o processo de seleção de um recurso para ser atribuído a uma tarefa, bem como a alocação pós-evento. Estes mecanismos foram explicados nos trabalhos de Callegari [CAL10] e Schlösser [SCH10].

Dessa maneira, foram desenvolvidos projetos de software hipotéticos, contendo recursos alocados às tarefas. Todos os casos de avaliação foram executados sobre um cenário composto por dez tarefas, sete recursos e seis papéis. Apesar do tamanho reduzido,é possível demonstrar as regras propostas pelo modelo SPIM (vide Capítulo 6). Quando uma condição específica é necessária, comentam-se quais foram as alterações necessárias para demonstrar algum evento. Também foram desenvolvidos fluxos organizacionais que serão associados a estes projetos de software.

Primeiramente, devem-se considerar os recursos que estão disponíveis para os cenários desta pesquisa. Os recursos são identificados com nomes próprios, neste caso: Mauricio, Géssica, Enzo, Mário, Fernando, Laura e José. Os papéis definidos foram Gerente de Projetos, Analista de Sistemas, Desenvolvedor, Testador, Integrador e Projetista de Banco de Dados. A Tabela 16 reúne os recursos e os papéis em que cada integrante do projeto pode assumir (também define se um papel possui perfil gerencial ou produtivo).

Tabela 16: Recursos e papéis disponíveis para o Projeto 1

Recursos	**Papéis**	**Perfil**
Mauricio	Gerente de Projetos	Gerencial
Géssica	Analista de Sistemas	Produtivo
Laura	Analista de Sistemas	Produtivo
Mário	Desenvolvedor, Testador	Produtivo
Enzo	Desenvolvedor	Produtivo
Fernando	Desenvolvedor, Integrador	Produtivo
José	Projetista de Banco de Dados	Produtivo

O cenário representa o Projeto 1 com as seguintes atividades: iniciação do projeto (INP), análise de requisitos (ANR), definição da arquitetura (DAR), modelagem do banco de dados (MBD), três componentes que são desenvolvidos em paralelo (DEV1, DEV2 e DEV3), integração dos componentes (INT), realização de testes dos componentes (RTC) e uma atividade de encerramento do projeto (ENP). Dessa forma, as tarefas receberam nomes curtos para facilitar a sua identificação. A Tabela 17apresenta as posições de trabalho das tarefas, o nome das tarefas, a abreviação destas tarefas, suas respectivas durações (em unidades discretas) e também suas dependências. Conforme nomenclatura adotada pelo modelo SPIM, uma atividade de projeto pode ser considerada como sendo gerencial ou produtiva (vide Capítulo 6).

Tabela 17: Tarefas do Projeto 1

Pos	**Tarefa**	**Abreviação**	**Tipo**	**Duração**	**Dependências**
1	Iniciação do Projeto	INP	Gerencial	2	-
2	Análise de Requisitos	ANR	Produtiva	6	1(INP)
3	Definição da Arquitetura	DAR	Produtiva	2	2(ANR)
4	Modelagem do Banco de Dados	MBD	Produtiva	3	3(ANR)
5	Desenvolvimento do Componente 1	DEV1	Produtiva	2	4(MDB)
6	Desenvolvimento do Componente 2	DEV2	Produtiva	4	4(MDB)
7	Desenvolvimento do Componente 3	DEV3	Produtiva	3	4(MDB)

8	Integração dos Componentes	INT	Produtiva	1	5, 6,7 (DEV1, DEV2, DEV3)
9	Realização de testes dos componentes	RTC	Produtiva	2	8 (INT)
10	Encerramento do Projeto	ENP	Gerencial	1	9 (RTC)

Os papéis relacionados às tarefas do Projeto1 são apresentados na Tabela 18.

Tabela 18: Informações das tarefas para o Projeto 1

Pos	Abreviação	Duração	Dependências	Papel
1	INP	2	-	Gerente de Projetos
2	ANR	6	1(INP)	Analista de Sistemas
3	DAR	2	2(ANR)	Analista de Sistemas
4	MBD	3	3(ANR)	Projetista de Banco de Dados
5	DEV1	2	4(MDB)	Desenvolvedor
6	DEV2	4	4(MDB)	Desenvolvedor
7	DEV3	3	4(MDB)	Desenvolvedor
8	INT	1	5, 6,7 (DEV1, DEV2, DEV3)	Testador
9	RTC	2	8 (INT)	Integrador
10	ENP	1	9 (RTC)	Gerente de Projetos

A Tabela 33 apresenta a configuração inicial das tarefas do Projeto 1, exibindo suas durações, seus tempos de início maiscedo (IMC), início mais tarde (IMT), término mais cedo (TMC), término mais tarde (TMT),unidades de folga, criticidade (ou seja, se faz parte do caminho crítico) e tempos atualmente programados para o início e fim de cada tarefa.

Tabela 19: Informações detalhadas das tarefas do Projeto 1

Pos	Tarefa	DUR	IMC	IMT	TMC	TMT	Folga	CRIT	Início	Fim	Dep.	Rec.
1	INP	2	1	1	2	2	0	S	1	2	-	Maurício
2	ANR	6	3	3	8	8	0	S	3	8	1	Géssica
3	DAR	2	9	9	10	10	0	S	9	10	2	Laura
4	MBD	3	11	11	13	13	0	S	11	13	3	José
5	DEV1	2	14	16	15	17	2	N	14	15	4	Fernando
6	DEV2	4	14	14	17	17	0	S	14	17	4	Enzo
7	DEV3	3	14	15	16	17	1	N	14	16	4	Mário
8	INT	1	18	18	18	18	0	S	18	18	5, 6,7	Fernando
9	RTC	2	19	19	20	20	0	S	19	20	8	Mário
10	ENP	1	21	21	21	21	0	S	21	21	9	Maurício

O cronograma correspondente para o Projeto 1 é mostrado na Figura 56.No topo da figura há uma régua de tempo que facilita visualizar a alocação de cada tarefa no tempo. Percebe-se, desta forma, que o projeto tem duração total de 21 unidades de tempo e que as tarefas DEV1 e DEV3 não são críticas (possuem folga). Assim, o projeto inicia com a tarefa INP no tempo 1, depois segue com a tarefa ANR, que ocupa os tempos de 3 a 8 com suas 6 unidades de duração, e assim sucessivamente.

Tarefa	Tempo																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
INP	1	2																			
ANR			1	2	3	4	5	6													
DAR									1	2											
MBD											1	2	3								
DEV1														1	2	-	-				
DEV2														1	2	3	4				
DEV3														1	2	3	-				
INT																		1			
RTC																			1	2	
ENP																				1	2

Figura 56. Cronograma do Projeto 1

A Figura 57 ilustra a rede de Petri gerada pelo protótipo SPIT para representar as atividades do cronograma do Projeto 1. Conforme ilustrado nesta figura, a ficha (*token*) permanece no lugar que representa a atividade Iniciação do Projeto (INP). O tempo de execução de cada atividade está representado nas transações desta rede de Petri. Assim, a legenda “T1(2)” informa que a transação “T1” deve demorar duas unidades de tempo para passar a ficha para o próximo lugar (Análise de Requisitos – ANR).

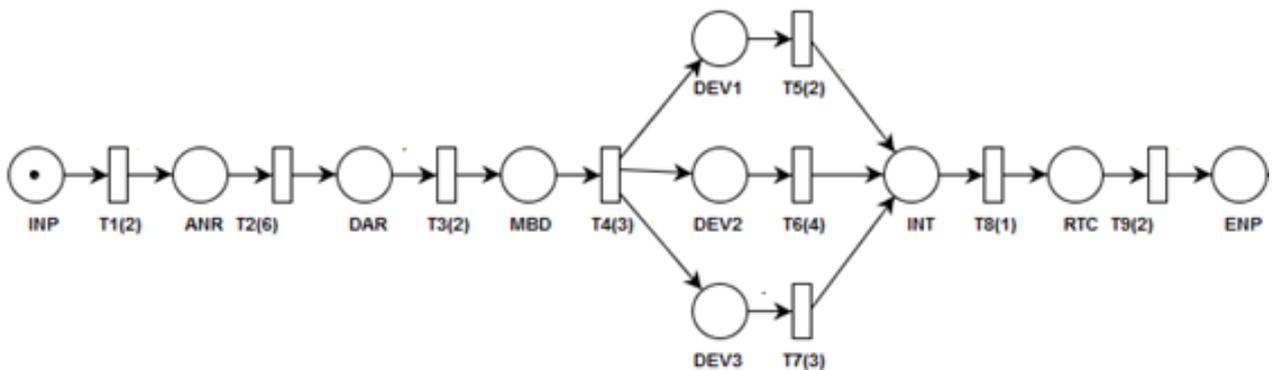



Figura 57. Rede de Petri para o cronograma do Projeto 1

O modelo SPIM, entretanto, foi concebido considerando a complexidade de identificar as relações de dependência entre as atividades dos projetos de software e os fluxos organizacionais (tais como: "aquisição de hardware", "contratação de novas pessoas", "configuração do ambiente de trabalho”). Cabe lembrar que os outros departamentos da organização são responsáveis por atualizar os prazos destes fluxos organizacionais (por exemplo, o departamento de recursos humanos é responsável pela gestão das atividades do fluxo organizacional “contratação de pessoas”). Assim, um dos objetivos do SPIM é auxiliar o gestor na antecipação das necessidades que surgem de diferentes fontes da organização através do envio de alertas que permite o replanejamento das atividades afetadas por esta relação de dependência. Para fins de ilustração, a Figura 58 apresenta o fluxo organizacional “contratação de pessoas”.Cabe relembrar que as atividades dos fluxos organizacionais são denominadas nesta pesquisa como atividades gerenciais de apoio. Neste fluxo organizacional, a atividade de “identificar as habilidades necessárias” é representada no lugar F1, a atividade “divulgar ofertas de vagas” é representado pelo lugar F2, a atividade “analisar candidatos” é representado pelo lugar F3, a atividade “contatarcandidatos selecionados” é representado pelo lugar F5, a atividade “avisar sindicato” é representado pelo lugar F4, a atividade “selecionar candidato” é representado pelo lugar F6 e a atividade “avisar gestor” é representado pelo lugar F7.

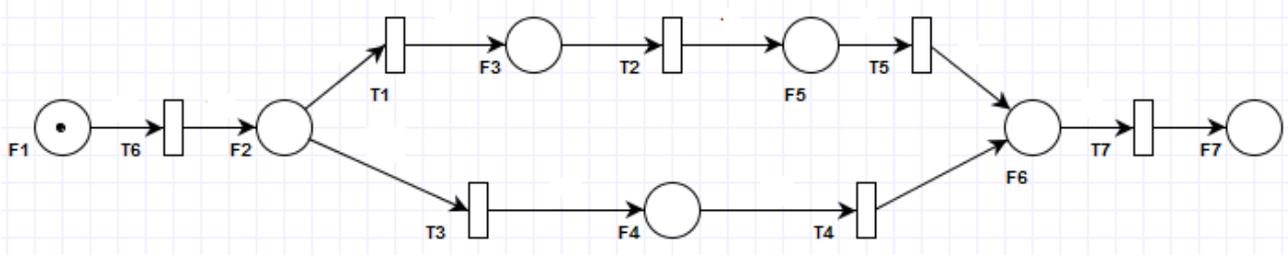



Figura 58. Fluxo Organizacional “contratação de pessoas”

A seguir serão criadas situações simuladas que geram eventos de reconfiguração sobre o Projeto 1. As consequências e o comportamento do protótipo para cada evento são também descritos.

7.2.2 Cenário 1

Para melhor ilustrar o uso das funcionalidades propostas para o SPIT, considere que o gestor de projetos deve acompanhar o desenvolvimento do Projeto 1 (vide Figura 58). Inicialmente, o gestor precisa realizar a configuração inicial do projeto através do

módulo de BackOffice do SPIT. Através desta interface ele pode registrar todas as informações especificadas no modelo SPIM, tal como a definição de papéis, produtos de trabalho e atividades básicas. Para simplificar este exemplo, objetivando facilitar o entendimento do SPIT neste cenário, considera-se que algumas informações de apoio ao SPIT já estão cadastradas na base de dados auxiliar (tal como o nome das fases do processo de desenvolvimento de projetos adotado pela organização, os processos gerenciais e áreas de conhecimento do PMBOK).

Considera-se, desta forma, que o gestor de projetos deve cadastrar apenas os dados essenciais para o entendimento deste cenário. Assim, aFigura 59 apresenta a interface de manutenção de projetos do módulo BackOffice. Esta interface apresenta campos para selecionar um programa e definir o nome, escopo, propósito, objetivos e custos indiretos do projeto. Cabe reforçar que o SPIT contém um módulo chamado SPIM Validator que funciona como um *add-in* da ferramenta comercial Microsoft Project. O SPIM Validator é responsável por verificar se o projeto está de acordo com as restrições definidas pelo modelo integrado SPIM. Dessa forma, a interface de manutenção de projetos do módulo de BackOffice do SPIT permite associar um projeto cadastrado na base de dados do protótipo SPIT com o arquivo utilizado por esta ferramenta comercial. Esta associação permite que as informações adicionais propostas pelo SPIM sejam exportadas para campos customizados da ferramenta comercial de gerência de projetos.

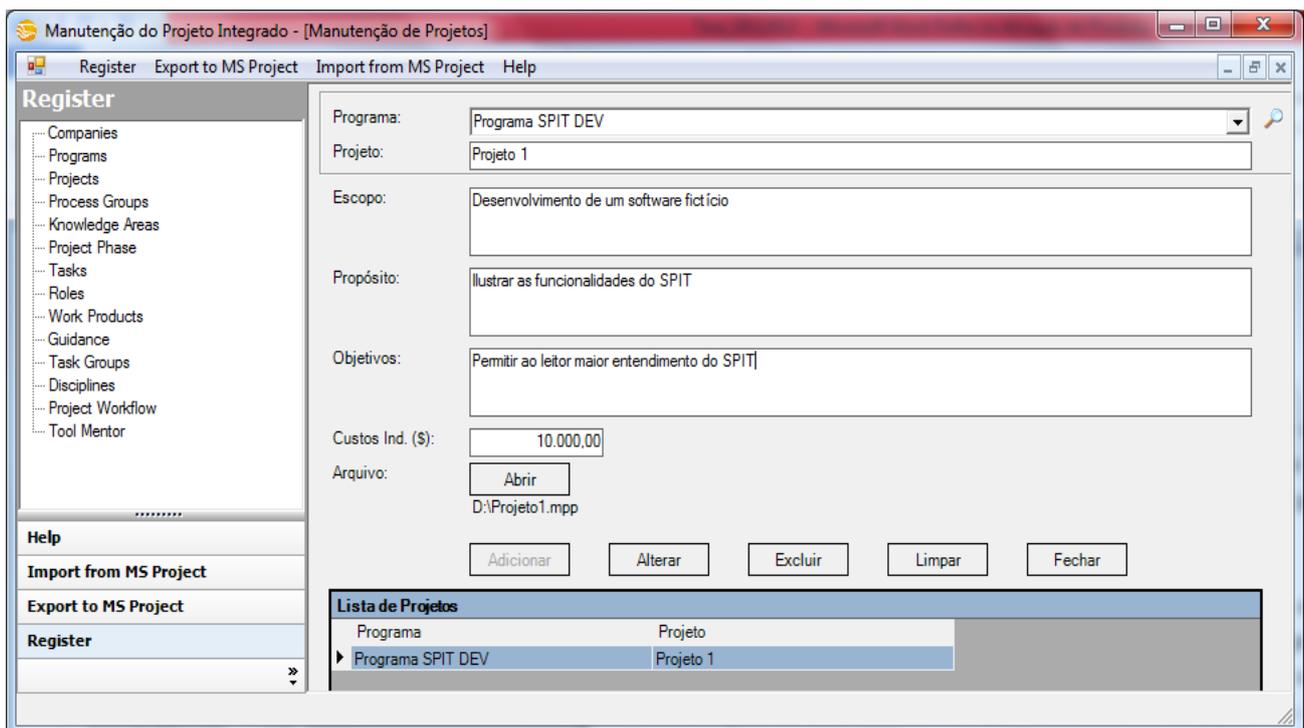



Figura 59. Interface de cadastro de projetos do BackOffice

Conforme observado na Tabela 16, esta reúne os recursos e os papéis em que cada integrante do projeto pode assumir. Os papéis adotados pela organização em seus projetos de software são registrados pelo módulo BackOffice do SPIT, conforme ilustrado na Figura 60. Conforme definido pelo modelo SPIM, os papéis são divididos em duas categorias: gerencial e produtivo.

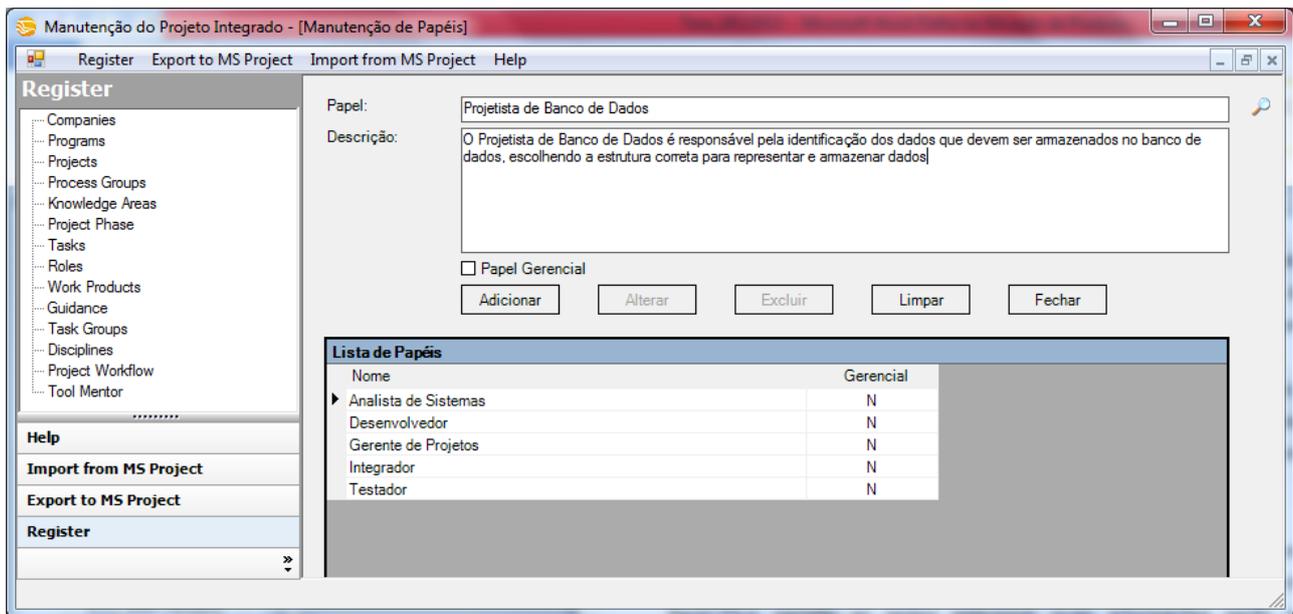



Figura 60. Interface de cadastro de papéis do BackOffice

Em seguida, o BackOffice apresenta uma interface para cadastrar os principais produtos de trabalho do projeto (vide Figura 61). Estes produtos também são classificados pelo modelo SPIM como gerenciais (tais como, ata de reunião e cronograma) e produtivos (tais como, diagramas de casos de uso, diagrama ER, etc). Além disso, pode-se informar se este produto de trabalho será criado ou não por uma empresa externa ao projeto.

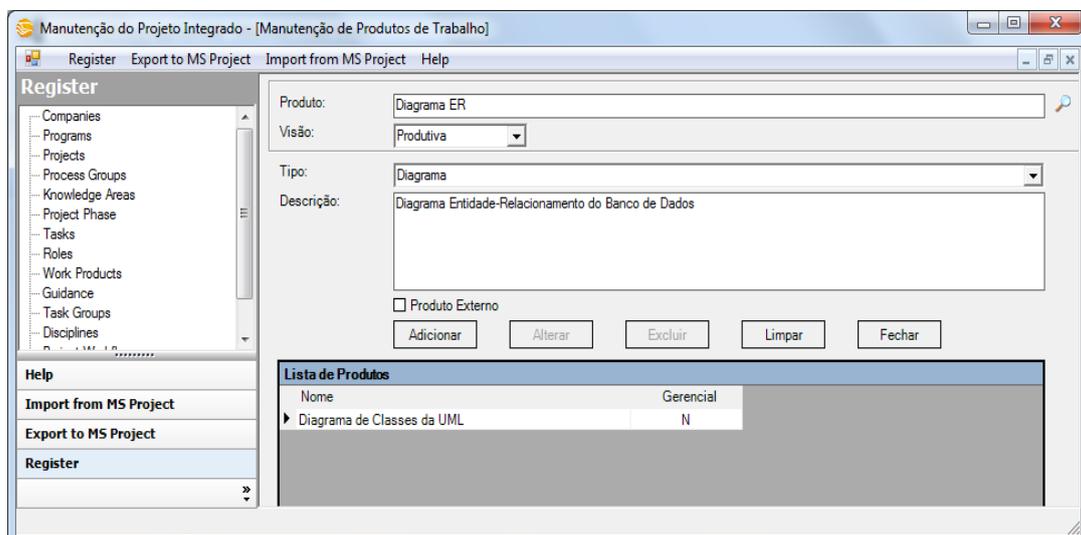



Figura 61. Interface de cadastro de produtos de trabalho do BackOffice

Uma vez cadastrado no banco de dados do SPIT, este tipo de informação deve ser exportado para um arquivo que possa ser utilizado pelo Microsoft Project. O módulo de BackOffice permite ao gestor selecionar quais informações serão exportadas para campos personalizados oferecidos por esta ferramenta comercial. A Figura 62 mostra o funcionamento da exportação dos papéis registrados na base de dados do SPIT para os campos personalizados do Microsoft Project.

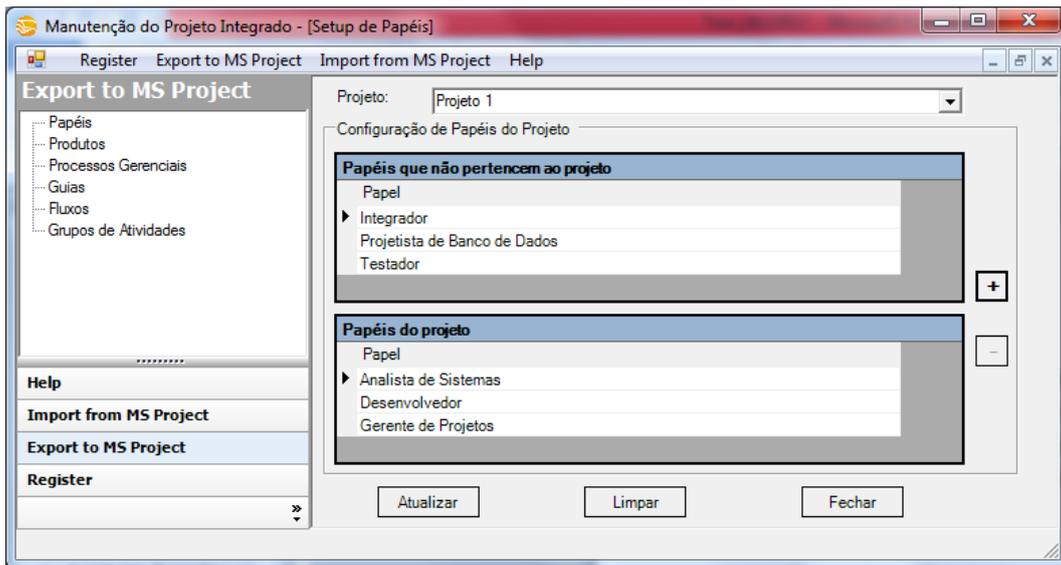



Figura 62. Exportação de Papéis para o Microsoft Project

Através da informação contida nestes campos personalizados, ilustrado na Figura 63, o gerente de projetos pode iniciar o registro dos itens do plano do projeto, tais como: tarefas, recursos e outras configurações. Durante o registro das atividades do projeto, o gestor deve informar, através dos campos personalizados, as seguintes informações necessárias para o modelo SPIM: identificador da atividade, tipo de atividade (produtiva, gerencial ou gerencial de apoio), associação com um ou mais fluxos de trabalho, associação com um processo gerencial do PMBOK, associação com um tipo de *guidance,* e associação com os produtos de trabalho que são criados, consultados ou modificados pela execução desta atividade. Para o registro dos recursos do projeto, o gestor deve informar se este recurso é gerencial ou produtivo.

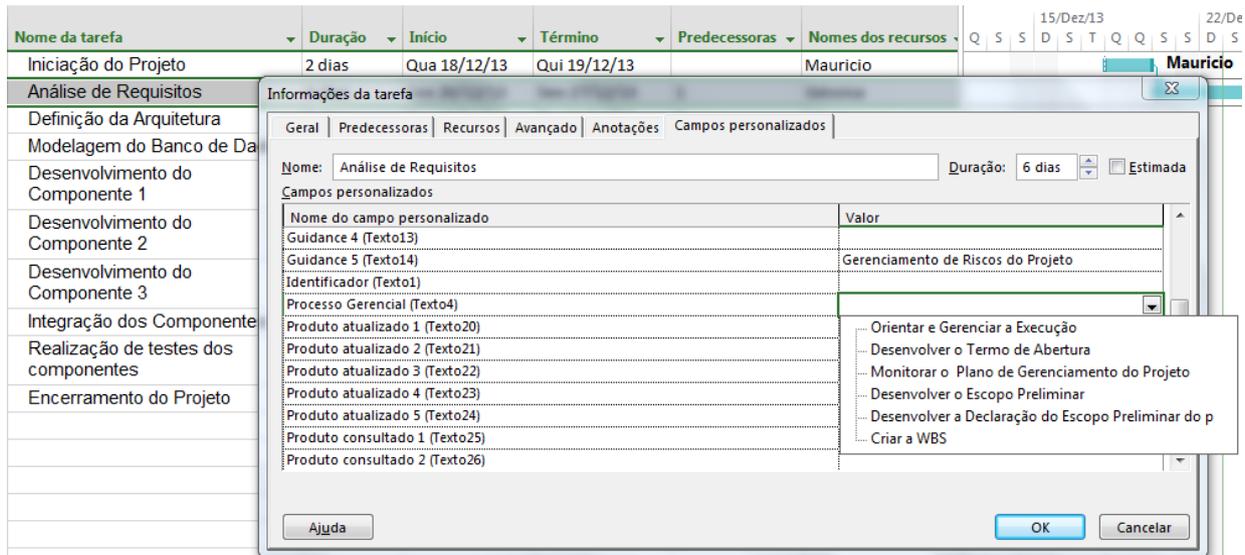



Figura 63. Visualização dos campos personalizadospara as atividades do projeto

Ao finalizar o cadastro das informações no Microsoft Project, o gestor deve obter um cronograma conforme ilustrado na Figura 64

		Modo da Tarefa	Nome da tarefa	Duração	Início	Término	Predecessoras	Nomes dos recursos
1			Iniciação do Projeto	2 dias	Qua 18/12/13	Qui 19/12/13		Mauricio
2			Análise de Requisitos	6 dias	Sex 20/12/13	Sex 27/12/13	1	Géssica
3			Definição da Arquitetura	2 dias	Seg 30/12/13	Ter 31/12/13	2	Laura
4			Modelagem do Banco de Dados	3 dias	Qua 01/01/14	Sex 03/01/14	3	José
5			Desenvolvimento do Componente 1	2 dias	Seg 06/01/14	Ter 07/01/14	4	Fernando
6			Desenvolvimento do Componente 2	4 dias	Seg 06/01/14	Qui 09/01/14	4	Enzo
7			Desenvolvimento do Componente 3	3 dias	Seg 06/01/14	Qua 08/01/14	4	Mário
8			Integração dos Componentes	1 dia	Sex 10/01/14	Sex 10/01/14	5;6;7	Fernando
9			Realização de testes dos componentes	2 dias	Seg 13/01/14	Ter 14/01/14	8	Mário
10			Encerramento do Projeto	1 dia	Qua 15/01/14	Qua 15/01/14	9	Mauricio

Figura 64. Cronograma Parcial do Projeto 1 no Microsoft Project

Considerando que todas as informações do Projeto 1 estão cadastradas nesta ferramenta comercial, o gestor pode realizar a validação do projeto em relação ao conjunto de restrições especificadas para o modelo SPIM (vide Tabela 8).A fim de ilustração, imagine que o gestor não informou o seguinte: que a atividade “Iniciação do Projeto” é do tipo produtiva e está associada ao recurso Maurício (que possui um papel gerencial) e que a atividade “Realização de testes dos componentes” é do tipo gerencial e está associada ao recurso Mário (que possui um papel produtivo).O modelo SPIM, através das restrições de números 8 e 9, considera que uma atividade gerencial deve ter pelo menos um papel gerencial como um de seus papéis e que uma atividade produtiva

deve ter pelo menos um papel produtivo como um de seus papéis. Considere também que o gestor associou o produto de trabalho produtivo à tarefa gerencial “Encerramento do Projeto”. As restrições 13 e 14 do modelo SPIM consideram que uma atividade gerencial não pode produzir ou modificar um produto de trabalho produtivo e que uma atividade produtiva não pode produzir ou modificar um produto de trabalho gerencial. Ainda, a tarefa gerencial “Encerramento do Projeto” foi associada a um guia produtivo (as restrições 23 e 24 do modelo SPIM tratam sobre esta situação).O SPIT, desta forma, percorre todas as informações contidas no projeto e alerta o gestor sobre os problemas encontrados (ilustrado na Figura 65).

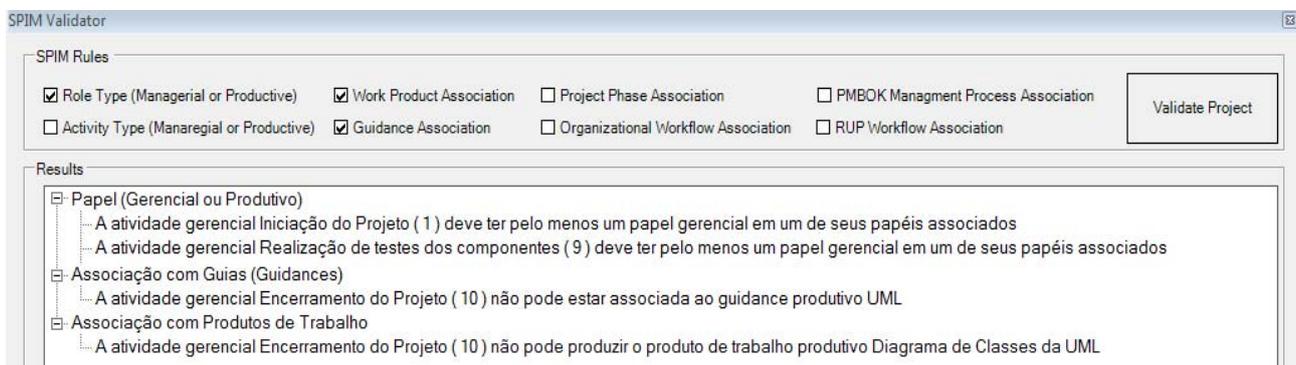



Figura 65. Validação do Projeto 1 pelo módulo SPIM Validator

O protótipo SPIT, conforme salientado anteriormente, implementa o conjunto de regras do modelo SPIM, fornecendo um conjunto de interfaces de suporte ao gerente de projetos durante o planejamento e a execução de projetos de software. Mais uma vez, o objetivo é facilitar o trabalho do gerente de projetos, que tem de levar em conta cada vez mais elementos em suas decisões e com cada vez menos tempo.

Os próximos cenários ilustram como o SPIM pode fornecer um suporte ao gerente de projetos para obter acesso às informações dos fluxos organizacionais e acompanhar, de maneira integrada, o andamento das atividades destes tipos de fluxos de trabalho.

7.2.3 Cenário 2

Supondo que o andamento do Projeto 1 tenha ocorrido conforme o planejado até o tempo 7. Entretanto, no tempo 8 o recurso Fernando deixou a empresa, gerando o evento ER3 descrito no Capítulo 4. A Figura 66 ilustra esta situação e destaca as tarefas queforam afetadas por esse evento (estas atividades foram comprometidas porqueo recurso Fernando estava originalmente alocado nelas).Consequentemente, o gestor deve reconfigurar o projeto de forma a encontrar substitutos para o recurso Fernando nestasposições de trabalho que ficaram sem recurso alocado.

Tarefa	Tempo																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
INP	1	2																			
ANR			1	2	3	4	5	6													
DAR									1	2											
MBD											1	2	3								
DEV1														1	2	-	-				
DEV2														1	2	3	4				
DEV3														1	2	3	-				
INT																		1			
RTC																			1	2	
ENP																				1	2

Figura 66. Cenário 1- recurso deixa a empresa

Depois de identificar as tarefas comprometidas, o modelo SPIM gera a rede de Petri que representa o cronograma do Projeto 1 (ilustrado anteriormente na Figura 56) através do *parser* denominado *ParserMSProject*. O comportamento do *parserParserMSProject* segue a lógica apresentada na Figura 67:

- Percorre todas as atividades do projeto
  - Verifica se esta atividade é representa uma tarefa simples, uma fase do projeto ou um agrupamento de tarefas
  - Caso represente uma tarefa simples
    - Cria um elemento *place* da rede de Petri
    - Verifica se esta atividade está ligada a outra atividade anterior. Neste caso, faz-se necessário criar os elementos *transitions* e *arcs* correspondentes.
      - Caso exista uma relação de precedência com outra atividade já mapeada:
        - » Cria o elemento *transition* que representa a relação de precedência entre as atividades
        - » Calcula o tempo de duração desta transição considerando os finais de semana e feriados
        - » Adiciona um elemento *arc* para conectar o elemento *place* (que corresponde à atividade dependente) com o elemento *transition*
        - » Adiciona um elemento *arc* para conectar o elemento *transition* com o elemento *place* (que corresponde à atividade predecessora)
  - Caso represente uma fase do projeto ou um agrupamento de tarefas
    - Segue para a próxima atividade do projeto

Figura 67. Comportamento do componente*ParserMSProject*

Para o cronograma ilustrado no Cenário 1, o algoritmo executado pelo componente *ParserMSProject* gera uma rede de Petri contendo 9 lugares, 8 transições e 18 arcos (vide Figura 68).Conforme mencionado anteriormente, o projeto está em andamento e o gestor se deparou com o evento de saída de um recurso no tempo 8 do cronograma

original (vide Figura 56). Desta forma, o componente *ParserMSProject*considera apenas as atividades do cronograma a partir do tempo 8 (neste exemplo simplificado, apenas a atividade INP não foi considerada na geração da rede de Petri).

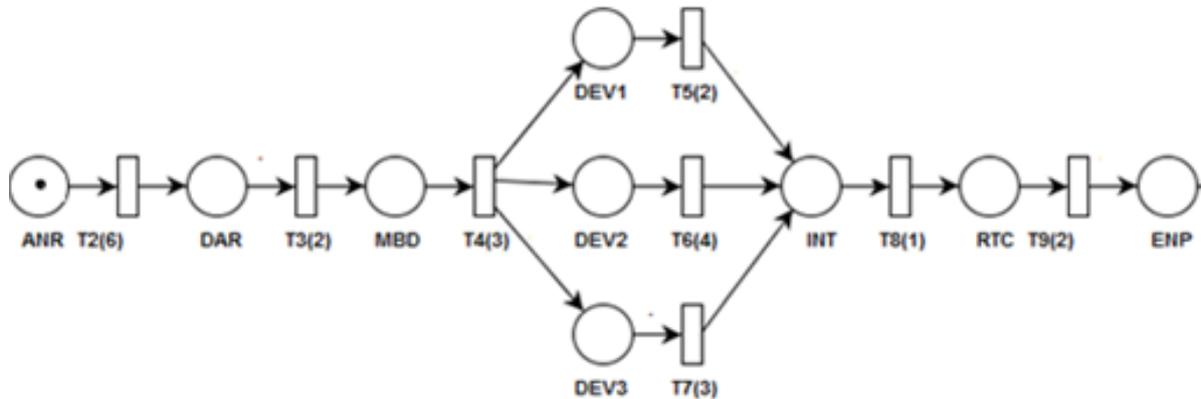



Figura 68. Rede de Petri gerada para o Cenário 1

Após a geração da rede de Petri, o modelo SPIM envia uma mensagem para o modelo de reconfiguração dinâmica proposto por Callegari [CAL10]. Este modelo dispara os *solvers*, os quais tentam localizar novos recursos que possam substituir o recurso Fernando. Neste cenário, pode-se considerar tanto o uso do *solver* padrão [CAL10] quando o uso do *solver* multiagentes [SCH10], uma vez que para esta pesquisa é necessário apenas considerar que o modelo retorne o nome de um recurso disponível.

Infelizmente, o modelo de reconfiguração retorna que nenhum recurso no *pool* possui disponibilidade no momento para assumir as tarefas afetadas. Portanto, o modelo SPIM instancia um fluxo organizacional para a contratação de recursos, antecipando que são necessárias, em média, 4 unidades de tempo para que a equipe de recursos humanos da empresa finalize esse processo. A pequena folga existente antes do fluxo organizacional (considerando que ainda restam 5 unidades de tempo antes que a primeira tarefa alocada para o recurso Fernando seja executada) indica que é possível iniciar o processo de contratação do novo recurso uma unidade de tempo antes do previsto.

Através da simulação da rede de Petri, o gestor pode avançar 4 unidades de tempo no cronograma. Desta forma, o protótipo SPIT simula uma mensagem de retorno do sistema de *workflow*responsável pela gestão do fluxo organizacional "contratação de pessoas". Assim, supondo que o fluxo organizacional retorne que o novo recurso (Maria) tenha sido contratado no tempo 12 para esse papel, o modelo SPIM envia uma requisição para o modelo de reconfiguração que automaticamente verifica a sua disponibilidade e o

atribui às tarefas DEV1 e INT. Desta forma, não há necessidade de gerar outras redes de Petri para simular possíveis deslocamentos de atividades no tempo.

Como observação final, se o fluxo organizacional enviasse uma resposta posterior ao tempo 12 (caso tivesse ocorrido apenas no tempo 15, por exemplo), o projeto seria atrasado de forma a terminar somente no tempo 23 (a Figura 69ilustra essa hipótese).

Tarefa	Tempo																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
INP	1	2																					
ANR			1	2	3	4	5	6															
DAR									1	2													
MBD											1	2	3										
DEV1														-	-	1	2	-	-				
DEV2											Fluxo Organizacional				-	1	2	3	4				
DEV3														-	-	1	2	3	-				
INT																				1			
RTC																					1	2	
ENP																						1	2

Figura 69. Execução do fluxo organizacional no Cenário 2

Neste caso, o protótipo SPIT tentaria gerar uma solução para o gestor antecipando as atividades INT e RTC (a primeira iniciando no tempo 14 e a segunda iniciando no tempo 15) e postergando as atividades DEV1, DEV2 e DEV3 (iniciando ao término da atividade RTC). A precedência entre as atividades do projeto seria modificada, conforme ilustra a Figura 70. Entretanto, o gestor dificilmente aceitaria esta solução, visto que é necessário primeiramente desenvolver o produto de software (através das atividades DEV1, DEV2 e DEV3) para depoisrealizar a integração de seus componentes (atividade INT) e, posteriormente, realizar ostestes (atividade RTC). Assim, não seria enviada uma CFP com esta configuração de atividades para o modelo de reconfiguração.

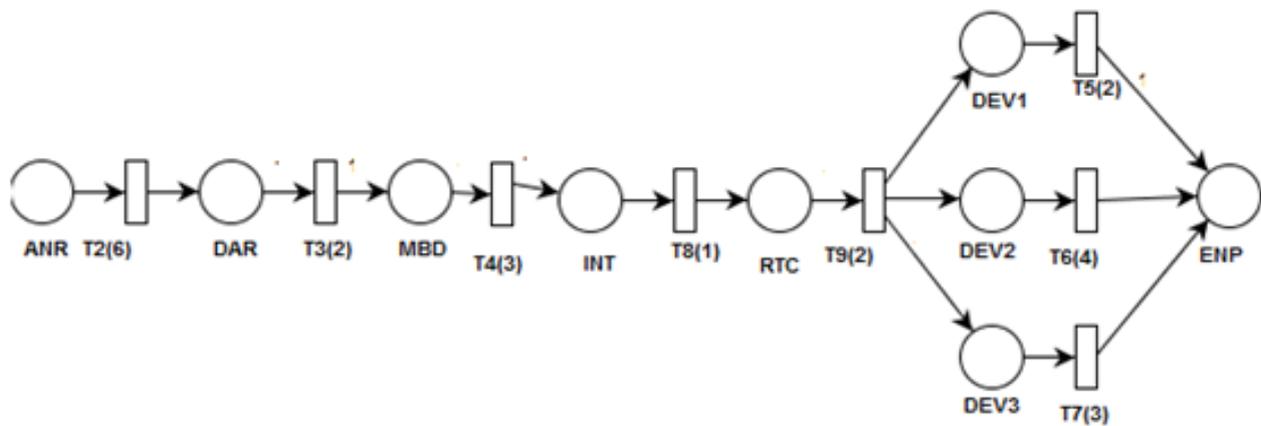



Figura 70. Rede de Petri para o Cenário 2 – atraso no fluxo organizacional

7.2.4 Cenário 3

Ao realizar o planejamento do Projeto 1, antes de iniciar a execução deste projeto, o gestor identificou a necessidade de aquisição de um servidor *web*. Assim, o gestor planejou entrar em contato com o setor de compras da organização no primeiro dia de execução deste projeto (mais especificamente, no tempo 1 do cronograma).

A execução do fluxo organizacional “adquirir materiais”, em resposta ao evento EA3 descrito no Capítulo 4, tem a duração de 10 unidades de tempo. Após, o gestor informa se alguma atividade do projeto depende do término da execução deste fluxo organizacional. Esta relação de dependência é possível através do campo customizado adicionado ao Microsoft Project chamado “Fluxo Organizacional”. Neste campo personalizado, o gestor adiciona uma relação de precedência entre atividades do projeto com atividades dos fluxos organizacionais. A Figura 71 ilustra esta situação e destaca as tarefas que foram afetadas por esse evento (estas atividades foram comprometidas porque o recurso servidor *web*está originalmente alocado nelas).

Tarefa	Tempo																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
INP	1	2																			
ANR			1	2	3	4	5	6													
DAR									1	2											
MBD											1	2	3								
DEV1														1	2	-	-				
DEV2														1	2	3	4				
DEV3														1	2	3	-				
INT																		1			
RTC																			1	2	
ENP																				1	2

Figura 71. Cenário 3- aquisição de servidor web

Após identificar as tarefas comprometidas, o modelo SPIM gera a rede de Petri que representa o cronograma do Projeto 1 (vide Figura 57). Conforme salientado anteriormente, o modelo SPIM foi desenvolvido considerando a complexidade de identificar as relações de dependência entre os projetos de software e os fluxos organizacionais. Assim, através de seu conjunto de restrições, o modelo SPIM identifica que a configuração atual do projeto resultará em um atraso no prazo do cronograma do Projeto 1. Considerando que o fluxo organizacional “adquirir materiais” consome 10 unidades de tempo e a atividade Definição da Arquitetura – DAR ocorre no tempo 9. Como resultado, a atividade DAR e todas suas atividades dependentes teriam seu tempo de início postergado em 1 unidade de tempo. A Figura 72 ilustra este atraso em uma unidade de tempo através do caractere *.

Tarefa	Tempo																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
INP	1	2																				
ANR			1	2	3	4	5	6														
DAR	Fluxo Organizacional								*	1	2											
MBD												1	2	3								
DEV1															1	2	-	-				
DEV2															1	2	3	4				
DEV3															1	2	3	-				
INT																			1			
RTC																				1	2	
ENP																					1	2

Figura 72. Execução do fluxo organizacional no Cenário 3

Pode-se observar, desta forma, que a dificuldade de identificar a distinção entre estes dois tipos de fluxos de trabalho pode resultar em distorções no planejamento deste projeto (as atividades gerenciais de apoio, pertencentes ao fluxo organizacional “adquirir materiais”, estão afetando negativamente no prazo das atividades deste projeto de software). Entretanto, o modelo SPIM possui um conjunto de regras (apresentadas no Capítulo 6) que auxilia o gerente de projetos na antecipação da identificação das atividades dos fluxos organizacionais, consequentemente, no replanejamento do projeto de software devido à existência de relações de dependência destas atividades com as atividades do projeto. Neste exemplo, para evitar que o Projeto 1 tenha alguma alteração nos prazos de suas atividades, o modelo SPIM informa ao gestor que ele deve entrar em contato com o departamento de compras da organização antes do início previsto do projeto (tempo zero). Assim, o gestor já deve fazer a solicitação do novo servidor web antes do início do projeto.

7.3 Limitações do Experimento

Especificamente sobre a realização do estudo experimental realizado, destacam-se as seguintes limitações:

- A primeira limitação deste experimento está relacionado com a quantidade de indivíduos que participaram, cerca de 36 pessoas. Apesar deste número de participantes ser inferior ao desejado, ele ainda é satisfatório para fins da pesquisa tendo em vista a avaliação analítica;
- Outra limitação está relacionada ao perfil dos participantes. Apesar dos esforços dos pesquisadores em aplicar este estudo experimental em empresas (inclusive, foi realizado uma palestra gratuita nos auditórios da PUCRS) somente alunos de pós-graduação em gestão de projetos se prontificaram a participar do experimento. Estes entrevistados, entretanto, apresentaram conhecimentos suficientes em gestão de projetos para que o experimento fosse realizado.

7.4 Considerações Finais do Capítulo

Este Capítulo apresentou as avaliações do modelo de reconfiguração proposto nesta tese, onde também foram detalhados os cenários usados para a geração dos dados e dos comportamentos obtidos como resposta. Conforme salientado anteriormente, foram desenvolvidos nesta tese de doutorado um modelo computacional e um protótipo de

ferramenta. Assim, as avaliações do modelo SPIM foram realizadas com o auxílio do protótipo desenvolvido.

A primeira avaliação foi realizada através em um estudo experimental considerando a proposta de planejamento integrado com o modelo SPIM em relação à proposta tradicional de planejamento de projetos de software. Seguindo esta abordagem, foi decidido que o objetivo desta pesquisa era de comparar a precisão e o esforço destes modelos de planejamento de projeto de software. Considerando a variável esforço, a avaliação mostrou que, estatisticamente, não havia diferença significativa para no planeamento de projetos de software utilizando o método tradicional em relação ao método SPIM. Entretanto, comparando os valores médios das duas abordagens, foi possível concluir que a precisão para realizar o planeamento usando o modelo SPIM foi maior do que usando o modelo tradicional. Cabe reforçar que para esta avaliação, participaram trinta e seis alunos de pós-graduação em Gestão de Projetos e foram utilizados cinco cenários simulando situações que ocorrem em projetos de software.

O estudo experimental, no entanto, possui uma abordagem quantitativa, de modo que um *Survey* também foi utilizado para avaliar os dados qualitativos.A fim de avaliar os conceitos advindos do modelo integrado SPIM sobre a sua aceitação e aplicabilidade, também foi realizada uma avaliação qualitativa exploratória. Os benefícios percebidos na realização do planejamento integrado de atividades gerenciais e produtivas podem ser visualizados na Tabela 15.

A segunda avaliação foi realizada de forma analítica, ou seja, verificando-se os resultados produzidos pelo modelo em função da ocorrência de eventos sobre cenários simulados. O objetivo desta avaliação foi de mostrar a integração do modelo SPIM com o modelo de referência para reconfiguração dinâmica de projetos de software proposto em Callegari [CAL10]. Todos os casos de avaliação foram executados sobre um cenário composto por dez tarefas, sete recursos e seis papéis. Apesar do tamanho reduzido, foi possível demonstrar algumas das principais regras propostas pelo modelo SPIM. Através deste tipo de avaliação, também foi possível demonstrar a integração proposta entre os projetos de software e os fluxos organizacionais através da geração de simulações com redes de Petri.

Finalmente, as conclusões e demais indicações de trabalhos futuros encontram-se no próximo Capítulo.

8 CONSIDERAÇÕES FINAIS

Esta tese de doutorado apresentou um modelo computacional para a reconfiguração dinâmica de projetos de software, considerando a integração das atividades dos projetos com os fluxos organizacionais. Inicialmente, realizou-se uma análise do estado da arte das soluções que atacam os problemas relacionados à reconfiguração dos projetos e se propôs, uma vez identificadas as atuais lacunas, um modelo capaz de suportar a integração de projetos de software com atividades pertencentes aos fluxos organizacionais. Modelos intermediários, oriundos da revisão da literatura, formaram a base conceitual sobre a qual o modelo final foi construído. As características do ambiente integrado desenvolvido para o planejamento de projetos de software, chamado SPIT, também foram descritas neste documento.

Conforme salientado anteriormente, os projetos de software envolvem incertezas e estão sujeitos a frequentes ajustes conforme as atividades sofrem alterações em resposta a muitos eventos (internos e externos ao projeto). Além disso, as empresas costumam possuir um conjunto de atividades (fluxos organizacionais) que podem ser compartilhados entre dois ou mais projetos simultâneos. Alguns exemplos são: o fluxo de trabalho para a requisição de viagens de trabalho dos empregados ou o fluxo para a aquisição de hardware e software necessários para a realização de um projeto. O desrespeito com as dependências entre as atividades desses diferentes fluxos de trabalho podem resultar em distorções no planejamento de projetos de software. Cabe lembrar que as atividades referentes aos fluxos de trabalho organizacionais usam recursos que não são atribuídos diretamente ao projeto de software. No entanto, esses recursos podem influenciar tanto em termos de prazos de atividades e custos do projeto de software (por exemplo, se o médico responsável pelo exame de admissão precisa se afastar da empresa por alguns dias, esse atraso pode afetar negativamente o cronograma dos projetos de software desta empresa). Conforme observado no item “3.2 Necessidade de Integração”, foram identificadas as principais carências de integração dos projetos com os fluxos organizacionais, ou seja: (1) Acesso às informações pertencentes aos outros departamentos da organização; (2) Identificação das relações de dependência entre as atividades dos fluxos organizacionais e dos projetos de software; (3) Capacidade de

minimizar distorções no planejamento de projetos (tais como, o aumento dos custos e atrasos nos prazos do projeto); e (4) Auxílio na identificação e mensuração dos custos indiretos do projeto. Assim, espera-se que uma solução satisfatória para o problema deva fornecer algum tipo de suporte para esta integração das atividades do projeto com os fluxos de trabalho organizacionais da empresa.

Dessa forma, foram apresentados os resultados de uma revisão sistemática sobre reconfiguração dinâmica de projetos de software. Nesta etapa da pesquisa, foi examinada a capacidade destes trabalhos relacionados em resolver 10 questões relacionadas com o tema desta pesquisa. Devido ao pequeno número de documentos retornados pelos motores de busca, nota-se que é possível realizar alguns comentários sobre a reconfiguração dinâmica de projetos, considerando a integração do gerenciamento de projetos com fluxos organizacionais. Mesmo os trabalhos mais recentes não apresentaram uma solução que atenda todos os problemas ao mesmo tempo. Uma razão pode ser que este seja um assunto que não tenha sido profundamente abordado por outros pesquisadores.

Analisando a literatura atual, não foi possível identificar estudos que abordam intimamente este assunto nem soluções atuais que forneçam algum tipo de integração entre projetos de software com os fluxos organizacionais. A identificação da interdependência dos fluxos de trabalho da empresa e do projeto de software durante o planejamento do projeto não é uma tarefa fácil. De acordo com os resultados nesta revisão sistemática, podemos dizer que o problema apresentado na questão foco (vide item “4.3.1 - Planejamento da Revisão Sistemática”) ainda não foi totalmente respondido pelas abordagens atuais.

As informaçõesobtidas nesta revisão da literatura, entretanto, foram suficientes para prosseguir com este trabalho, uma vez que estes fatos indicaram a necessidade de novas investigações e novas soluções para esta área. Consequentemente, aprofundaram-se os estudos sobre o desenvolvimento do modelo integrado SPIM. Paralelamente, novas funcionalidades foram adicionadas ao software SPIT.

Em seguida, foi apresentado o planejamento e a execução de dois estudos experimentais, realizados com estudantes de graduação e pós-graduação, que analisou dois métodos distintos de gestão de projetos (tradicional e SPIM). Cinco cenários foram elaborados com o objetivo de coletar informações para comparar a precisão e o esforço utilizando o planejamento tradicional e o planejamento integrado SPIM. Este experimento

revelou que o uso do modelo SPIM ajudou os gestores na criação e condução de um plano de projeto mais preciso do que com uso do método tradicional. Muitas vezes, o gerente de projetossomente percebe a necessidade de entrar em contato com outro departamento para obter alguma informação apenas no momento em que a equipe deve executar a atividade do projeto. A obscuridade em identificar este tipo de relacionamento durante o planejamento e execução de um projeto de software pode afetar negativamente o cronograma do projeto. Esta prova foi claramente observada durante este estudo empírico ao analisar a variável precisão.

Evidências relacionadas com a variável esforço também puderam ser extraídas neste estudo: o tempo para planejar as atividades do projeto utilizando o modelo SPIM é semelhante ao do modelo tradicional. A idéia por trás do modelo SPIM vem da necessidade de reduzir a complexidade em visualizar as interdependências de ambos os fluxos de trabalho organizacionais e do fluxo de atividades de um projeto individual. A maior parte do esforço em usar o modelo SPIM está relacionada com o preenchimento das informações extras propostas por este modelo. No entanto, os resultados da variável esforço não se tornou favorável para o método tradicional.

Outra funcionalidade desenvolvida para o protótipo SPIT é a geração de uma rede de Petri a partir da WBS do projeto (pelo módulo *Compiler*). Rede de Petri é uma ferramenta para descrever e estudar sistemas de processamento de informações que é caracterizada como sendo concorrente, assíncrona, distribuída, paralela, não determinística e/ou estocástica [MUR89]. Através do mapeamento do diagrama de rede que representa a WBS para o a rede de Petri, foi possível simular três cenários de reconfiguração de projetos (em resposta a eventos externos).

O modelo de simulação dos projetos de software desenvolvido nesta pesquisa utilizou como referência o modelo de programação preditiva-reativa. Este é um processo de programação e reprogramação do projeto onde as atividades são revistas em resposta a eventos ocorridos em tempo real. Assim, considerando as diferentes estratégias de reprogramação das atividades do projeto pesquisadas, a solução proposta utilizou estratégia de reparo/ajuste da programação do projeto. Cabe relembrar que esta estratégia limita o escopo a pequenos ajustes da programação atual, enquanto que a estratégia da programação completa gera uma nova programação das atividades a partir do zero. Conforme observado no Capítulo 4, a estratégia da reprogramação completa pode resultar em instabilidade e falta de continuidade do cronograma, levando a produção

adicional de custos (atribuídos ao nervosismo da equipe). Ainda, a política adotada para identificar quando reprogramar as atividades do projeto foi a de reescalonamento orientada a eventos (que é acionada em resposta a um acontecimento inesperado que altera o status atual do projeto).

Finalmente, o modelo SPIM adota a heurística de deslocamento para a direita (*right-shift*) para propor o deslocamento das atividades no cronograma. Desta forma, a solução desenvolvida desloca o tempo das tarefas restantes da programação para frente, considerando a quantidade de tempo ocioso dos recursos envolvidos. A heurística da reparação parcial programação também é adotada, uma vez que esta reagenda apenas as tarefas diretamente e indiretamente afetadas pela perturbação causada por algum evento (de modo a minimizar, tanto o aumento da eficiência quanto do desvio da programação inicial).

8.1 Contribuições

Destacam-se como principais contribuições desta tese:

- a elaboração de modelos intermediários que abordaram a integração do PMBOK com o os processos de desenvolvimento de software SPEM, RUP e OPEN;
- o desenvolvimento do modelo de integração entre a gestão de projetos e os fluxos organizacionais (denominado de modelo SPIM) com base nos modelos de referência PMBOK, RUP, OPEN e SPEM 2.0;
- a integração do modelo SPIM com os conceitos propostos pelo modelo de reconfiguração dinâmica de projetos de software proposto por Callegari [CAL10].
- a adequação dos conceitos propostos pela integração do modelo SPIM considerando o uso da arquitetura multiagentes proposta por Schlösser [SCH10];
- o desenvolvimento do protótipo SPIT, visando demonstrar os conceitos propostos pelo modelo integrado SPIM e pelo seu conjunto de restrições;
- a avaliação do modelo através de um estudo experimental com alunos de pós-graduação em gestão de projetos;
- a avaliação da integração do modelo SPIM com o modelo de reconfiguração através de cenários de situações-problema típicas;

8.2 Publicações Relacionadas à Tese

Durante o desenvolvimento da tese, foram realizadas as seguintes publicações.

Tabela 20: Publicações relacionadas à tese

#	Local	Título	Autores
1	ICIS 2012 - International Conference on Information Systems	A Systematic Review on the Integration of Project Management with Organizational Flows	Maurício Covolan Rosito Ricardo Melo Bastos
2	*International Journal of Computer and Communication Engineering*	*Project Management and Software Development Processes: Integrating PMBOK and OPEN*	Maurício Covolan Rosito Daniel Antonio Callegari Ricardo Melo Bastos
3	*IADIS 2012 - International Conference Applied Computing*	*An Integrated Environment for Software Project Planning*	Maurício Covolan Rosito Ricardo Melo Bastos
4	*ISDA 2012 - International Conference on Intelligent Systems Design and Applications*	*A Model to Integrate Software Project Management with Organizational Workflows*	Maurício Covolan Rosito Ricardo Melo Bastos
5	*ICEIS 2013 - International Conference on Enterprise Information Systems*	*An Experimental Study on the Dynamic Reconfiguration of Software Projects*	Maurício Covolan Rosito Ricardo Melo Bastos
6	*International Journal of Computer Information Systems and Industrial Management Applications*	*SPIM: An Integrated Model of Software Project Management and Organizational Workflows*	Maurício Covolan Rosito Ricardo Melo Bastos

Para auxiliar no avanço da investigação na área, sugerem-se algumas alternativas de continuidade ao trabalho.

8.3 Indicação de Trabalhos Futuros

Durante a elaboração desta tese, algumas idéias adicionais foram identificadas e são aqui indicadas como sugestão para trabalhos futuros:

- formalização das restrições do metamodelo através da linguagem OCL;
- implementação de algum tipo de validação léxica e estrutural nos *parsers* do módulo Compiler do SPIT.
- desenvolvimento de um protocolo para a avaliação do modelo proposto com empresas de software, utilizando o protótipo em projetos reais;
- integração com fluxos organizacionais complexos;

- integração entre as plataformas de desenvolvimento do modelo SPIM com o modelo de referência proposto por Callegari [CAL10] e por Schlösser [SCH10];

A continuidade desta pesquisa indica novas contribuições para engenharia de software, melhorando nossa compreensão dos aspectos relacionados à reconfiguração dinâmica de projetos, considerando os relacionamentos da gerência de projetos com os fluxos organizacionais.

REFERÊNCIAS

[ABU97] ABUMAIZAR, R.J., SVESTKA, J.A.“Rescheduling job shops under random disruptions”. *International Journal of Production Research*, vol. 35-7,1997, pp. 2065–2082.

[ADB91] ABDEL-HAMID, T., MADNICK, S.E. “Software Project Dynamics: An Integrated Approach”. Englewood Cliffs, NJ, Ed. Prentice Hall, 1991, 264p.

[ARA09] ARAUZO, J.A., GALÁN, J.M., PAJARES, J., PAREDES, A.L. “Online Scheduling in Multi-projects Environments: A Multi-Agent Approach”. In: 7th International Conference on Practical Applications of Agents and Multi-Agent Systems – PAAMS, 2009, pp. 293-301.

[AYE05] AYE, T., TUN, K.“A Collaborative Mobile Agent-based Workflow System”. In: 6th Asia-Pacific Symposium on Information and Telecommunication Technologies, 2005, pp 59-65.

[AYT05] AYTUG, H., LAWLEY, M. A., MCKAY, K., MOHAN, S., UZSOY, R. “Executing production schedules in the face of uncertainties: A review and some future directions”. *European Journal of Operational Research*, vol. 161-1, 2005, pp. 86–110.

[BAJ07] BAJEC, M., VAVPOTIC, D., KRISPER, M. “Practice-Driven Approach for Creating Project-Specific Software Development Methods”, *Information and Software Technology*, 2007, pp. 345-365.

[BEC99] BECK, K. “Extreme Programming Explained: Embrace Change”, Addison Wesley, 1999.

[BEL96] BELZ, R., MERTENS, P. “Combining knowledge-based systems and simulation to solve rescheduling problems”. *Decision Support Systems*, vol 17-2, 1996, pp. 141–157.

[BEN07] BENDOLY, E., SWINK M. “Moderating effects of information access on project management behavior, performance and perceptions”. *Journal of Operations Management*, vol. 25-3, 2007, pp.604-62.

[BEN09] BENCOMO, A. “Extending the RUP”. Capturado em: http://www.ibm.com/developerworks/rational/library/05/323_extrup1/index.html, Novembro 2006.

[BEZ07] BEZERRA, E. “Princípios de Análise e Projeto de Sistemas com UML”. Rio deJaneiro: Elsevier, 2007, 2ª Edição, 369p.

[BIL03] BILLINGTON, J., CHRISTENSEN, S., VAN HEE, K. E., KINDLER, E., KUMMER, O., PETRUCCI, L., POST, R., STEHNO, C., WEBER, M.: “The Petri Net Markup Language: Concepts, Technology, and Tools”. In: W. M. P. van der Aalst and E. Best, eds., Applications and Theory of Petri Nets 2003, 24th International Conference, Eindhoven, The

Netherlands. 2003.

[BIO05] BIOLCHINI, J., MIAN, P.G., NATALI, A.C.C., TRAVASSOS, G.H. “Systematic review in software engineering”. Rio de Janeiro, 2005, 31p.

[BON00] BONGAERTS, L., MONOSTORI, L., MCFARLANE, D., KADAR, B. “Hierarchy in distributed shop floor control”. *Computers in Industry*,vol. 43-2, 2000,pp.123–137.

[BOO00] G. BOOCH, J. RUMBAUGH, I. JACOBSON. “UML, guia do usuário”. Rio de Janeiro: Elsevier, 2000, 472 p.

[BRAN99] BRANDIMARTE, P., VILLA, A. “Modelling manufacturing systems: from aggregate planning to real-time control”. Berlin: Springer, 1999.

[BRE01] BRENNAN, R.W., NORRIE, D. H. “Evaluating the performance of reactive control architectures for manufacturing production control”. *Computers in Industry*, vol. 46-3, 2001, pp. 235–245.

[CAL07] CALLEGARI, D. E BASTOS, R. “Project Management and Software Development Processes: Integrating RUP and PMBOK”. In: ICSEM - International Conference on Systems Engineering and Modeling,Haifa, Israel, 2007, 8p.

[CAL08] CALLEGARI, D., ROSITO, M., BLOIS, M., BASTOS. R. “An Integrated Model for Managerial and Productive Activities in Software Development”. In: ICEIS - 10th International Conference on Enterprise Information Systems, Spain, 2008, 8p.

[CAL10] CALLEGARI, D. “Reconfiguração Dinâmica de Projetos de Software: Um modelo para alocação de recursos e programação de atividades em ambientes multiprojetos com recursos compartilhados”. Tese de Doutorado, PUC-RS, Porto Alegre, 2010.

[CAR97] CARDOSO, J., VALETTE, R. “Redes de Petri”. Florianópolis: Ed. UFSC, 1997.

[CHA10] CHATFIELD, C., JOHNSON, T. “Microsoft Office Project 2010 Step by Step”. Redmond: Microsoft Press, 2010.

[CHE05] CHEN, Y., FANG, M. “Research on Resource Scheduling for Development Process of Complicated Product”. In: Proceedings of the 9th International Conference on Computer Supported Cooperative Work. Coventry University, UK, vol. 1, 2005, pp.229-233.

[CHU92] CHURCH, L. K., UZSOY, R. “Analysis of periodic and eventdriven rescheduling policies in dynamic shops”. *International Journal of Computer Integrated Manufacturing*, vol. 5-3, 1992, pp. 153–163.

[COE03] COELHO, C. “Um Modelo para Adaptação de Processos de Software”, Dissertação de Mestrado. Universidade Federal de Pernambuco, 2003.

[COW00] COWLING, P. I., OUELHADJ, D., PETROVIC, S. “Multi-agent systems for dynamic scheduling”. In: Proceedings of the nineteenth workshop of planning and scheduling of the UK, UK, 2000, pp. 45–54.

[COW02] COWLING, P. I., JOHANSSON, M. “Using real-time information for effective dynamic scheduling”. *European Journal of Operational*

Research, vol. 139-2, 2002, pp. 230–244.

[DAN95] DANIELS, R. L., & KOUVELIS, P.“Robust scheduling to hedge against processing time uncertainty in single-stage production”. *Management Science*, vol. 41-2, 1995,pp. 363–737.

[DEB07] DEBLAERE, F., DEMEULEMEESTER, E., HERROELEN, W., VAN DE VONDER, S. “Robust Resource Allocation Decisions in Resource-Constrained Projects”. *Decision Sciences*, vol. 38, 2007, pp. 5-37.

[DEL99] DELEN, D., BENJAMIN, P., ERRAGUNTLA, M.“An Integrated Toolkit for Enterprise Modeling and Analysis”.In: Proceedings of the 1999 Winter Simulation Conference, 1999, pp. 289-297.

[DER99] DERNIAME, J. C., KABA B. A., WARBOYS B. “Software Process: Principles, Methodology, and Technology”.Lecture Notes in Computer Science 1500, Springer, 1999.

[DEW08] DEWSON, R.“Beginning SQL Server 2008 for Developers: From Novice to Professional”. Apress, 2008.

[DIW08] DIWAKAR, D. E DIWAKAR, S.“CINWEGS-An Integrated Web and Grid Services Framework for Collaborative Solutions”. In: 4th International Conference on Next Generation Web Services Practices, 2008, pp. 21-27.

[DON08] DONG, F., LI, M., ZHAO, Y., LI, J., YANG, Y. “Software Multi-Project Resource

Scheduling: A Comparative Analysis”. In: Lecture Notes in Computer Science (Making Globally Distributed Software Development a Success Story), Berlin: Springer Link, 2008, vol. 5007, pp. 63-75.

[DOR95] DORN, J., KERR, R. M., THALHAMMER, G. “Reactive scheduling: improving the robustness of schedules and restricting the effects of shop floor disturbances by fuzzy reasoning”. *International Journal of Human Computer Studies*, vol. 42, 1995, pp. 687–704.

[DUT90] DUTTA, A. “Reacting to scheduling exceptions in FMS environments”. *IIE Transactions*, vol. 22-4, 1990, pp. 33–314.

[ENG03] ENGWALL, M., JERBRANT A. “The resource allocation syndrome: the prime challenge of multi-project management?”. *International Journal of Project Management*, vol. 21-6, 2003, pp. 403-409.

[EVE00] EVERS, J.H. “Multi-project support issues: cycle time and schedule effects when people support multiple concurrent projects”. In: Proceedings of the 2000 IEEE Engineering Management Society, 2000, pp. 19-24.

[FAL98] FALBO, R. A., “Integração de Conhecimento em um Ambiente de Desenvolvimento”, Tese de Doutorado, COPPE/UFRJ, 1998.

[FEN97] FENTON N., PLEEGER S.L. “Software Metrics: A rigorous & Pratical Approach”. Boston, MA: PWS Publishing Company, 2nd edition, 1997.

[FIE05] FIELD, A. “Discovering Statistics Using SPSS”. Sage Publications, 2nd edition, 2005.

[FIR02] FIRESMITH, D., HENDERSON-SELLERS, B. “The OPEN Process Framework – An Introduction”, Addison-Wesley: Harlow, 2002.

[FRI00] FRICKE, S.E, SHENHAR, A.J.“Managing multiple engineering projects in a manufacturing support environment”. *IEEE Transactions on Engineering Management*, vol. 47-2, 2000, pp. 258-268.

[FUG00] FUGGETTA, A., “Software Process: A Roadmap”. In: Future of Software engineering Limerick Ireland, ACM, 2000.

[GAR94] GARNER, B. J., RIDLEY, G. J. “Application of neuronal network process in reactive scheduling”. In:Knowledge-based reactive scheduling, 1994, pp. 19–28.

[GAR95] GARETTI, M., TAISCH, M.“Using neuronal networks for reactive scheduling”. In: Artificial intelligence in reactive scheduling, 1995, pp. 146–147.

[GIN95] GINSBERG, M. P., QUINN, L. H. “PROCESS Tailoring and the Software Capability Maturity Model.” Technical Report, November 1995.

[GLO95] GLOVER, F., KELLY, J. P., & LAGUNA, M.“Genetic algorithms and tabu search: hybrids for optimization”. *Computers of Operation Research*, vol. 22-1, 1995,pp. 111–134.

[GOU98] GOU, L., LUH, P. B., KYOYA, Y. “Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation”. *Computers in Industry*,vol. 37-3, 1998,pp. 213–231

[GRA97] GRAHAM, I., HENDERSON-SELLERS, B., YOUNESSI, H. “The OPEN Process Specification”. New York: ACM Press, 1997, 314 p.

[GRU69] GRUBBS, F. E. “Procedures for detecting outlying observations in samples”. *Technometrics*, vol. 11, 1969, pp. 1-21.

[GRU04] GRUDIN, J.“Managerial Use and Emerging Norms: Effects of Activity Patterns on Software Design and Deployment”. In: Proceedings of the 37th Hawaii International Conference on System Sciences, 2004, pp. 1-10.

[HEL03] HELLER, M., WESTFECHTEL, B. “DYnamic Project and Workflow Management for Design Processes in Chemical Engineering”. Process Systems Engineering, 2003.

[HEC00] HENNING, G. P., CERDA, J. Knowledge-based predictive and reactive scheduling in industrial environments. *Computers and Chemical Engineering*, vol. 24-9, 2000, pp. 2315–2338.

[HEN00] HENDERSON-SELLERS, B., DUÉ, R. GRAHAM, I., COLLINS, G. “Third generation OO processes: a critique of RUP and OPEN from a project management perspective”.In: Seventh Asia-Pacific Software Engineering Conference, 2000. pp. 428-435.

[HEN01] HENDERSON-SELLERS, B., COLLINS, G., DUÉ, R., GRAHAM, I.M. “A Qualitative Comparison of Two Processes for Object-Oriented Software Development”. *Information and Software Technology*, United

Kingdom: Elsevier Science, vol. 43, November 2001, pp. 705-724.

[HER05] HERROELEN, W., LEUS, R. “Project scheduling under uncertainty: Survey and research potentials”. *European Journal of Operational Research*, vol. 165-2, 2005, pp. 289–306

[HES09] HU, H., LI, Z.“Modeling and scheduling for manufacturing grid workflows using timed Petri nets”. *The International Journal of Advanced Manufacturing Technology*, vol. 42, 2009, pp. 553–568.

[HIG08] HIGGINS, J.P., GREEN S. “Cochrane Handbook for Systematic Reviews of Interventions Version 5.0.1”.The Cochrane Collaboration, 2008.

[HUM89] HUMPHREY, W.S. “Managing the Software Process”, Addison-Wesley, 1989.

[HUM91] HUMPHREY, W.S., SNYDER, T.R. AND WILLIS, R.R., “Software Process Improvement at Hughes Aircraft”.In: Institute of Electrical and Electronic Engineers - IEEE, 1991, pp. 11-23.

[IBM11] IBM. “SPSS Statistical Analysis”. Capturado em:http://www-01.ibm.com/software/analytics/spss/products/statistics/, Julho 2011.

[JAC01] JACOBSON, I., BOOCH G., RUMBAUGH J. “The Unified Software Development Process”.Upper Saddle River, Addison Wesley, 2001.

[JAL04] JALOTE, P., PALIT, A., KURIEN, P., PEETHAMBER, V.T. “Ti pp.117-127meboxing: a process model for iterative software development”.*Journal of Systems and Software*, vol.70, 1-2, 2004.

[JEN01] JENSEN, M.T. “Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures”. *Applied Soft Computing,*vol. 1-1, 2001, pp. 35–52.

[JIA06] JIAMTHUBTHUGSIN, W., SUTIVONG, D. “Resource Decisions in Software Development Using Risk Assessment Model”. In: 39th Hawaii International Conference on Systems Sciences, 2006, 8p.

[JOS05] JOSLIN, D., POOLE, W. “Agent-Based Simulation For Software Project Planning”. In Proceedings of the 37th Conference on Winter Simulation. Orlando, Florida, 2005, pp 1059-1066.

[JOZ98] JOZEFOWSKA, J., MIKA, M., ROYCKI, R., WALIGORA, G., WGLARZ, J. W. “Local search meta-heuristics for discrete-continuous scheduling problems”.*European Journal of Operational Research*, vol. 107-2, 1998,pp. 354–370.

[JUN00] JUNGEL, M., KINDLER, E., WEBER, M.“Towards a Generic Interchange Format for Petri Nets - Position Paper-“. In: XML/SGML based Interchange Formats for Petri Nets, Aarhus, Denmark, 2000.

[JUR99] JURISON, J. “Software Project Management: The Manager’s View”. *Communications of the Association for Information Systems*, vol.2-3, Novembro 1999, 57p

[JUR03] JURISTO, N., MORENO, A. “Basics of Software Engineering Experimentation”. Kluwer academic Publishers. 2nd edition, 2003.

[KER95] KERR, R. M., SZELKE, E. “Artificial intelligence in reactive scheduling”. Dordrecht: Kluwer Academic, 1995.

[KER00] KERZNER, H.“Applied project management: best practices on implementation”. New York: Wiley e Sons, 2000, 534p.

[KIM95] KIM J.W., DESROCHERS A.A., SANDERSON A.C. “Task planning and project management using Petri nets”. Proceedings of the IEEE International Symposium on Assembly and Task Planning. Pittsburgh: IEEE, 1995, pp. 265-270.

[KIM05] KIM K.W., YUN Y.S., YOON J.M., ET AL.“Hybrid genetic algorithm with adaptive abilities for resourceconstrained multiple project scheduling”. *Computers in Industry*, vol. 56-2, 2005, pp. 143-160.

[KIT96] KITCHENHAM B. “Software Metrics”. Oxford: Blackwell Plublishers Inc, 1996.

[KIT04] KITCHENHAM, B. “Procedures for performing systematic reviews”. Joint Technical Report Software Engineering Group, Department of Computer Science, Keele University, 2004, 33p.

[KRU00] KRUCHTEN, P.“The Rational Unified Process: An Introduction”.Addison-Wesley, 2nd edition, 2000.

[KUM98] KUMAR A., GANESH L.S. “Use of Petri nets for resource allocation in projects”].*IEEE Transaction on Engineering Management*, vol. 45-1, 1998, pp. 49-56.

[LEE02] LEE, S., SHIM, J., WU, C. “A metal model approach using uml for task assignment policy in software process”. In: Proceedings of the Ninth Asia-Pacific Software Engineering Conference – APSEC, 2002.

[LEE04] LEE, B., MILLER, J. “Multi-Project Management in Software Engineering Using Simulation Modelling”. *Software Quality Journal,*vol. 12, 2004, pp.59-82.

[LEA04] LEACH, L. P. “Critical Chain Project Management”, Artech House Publishers,2004, 2a. edição, 263p.

[LEE06] LEE, J., LEE, N.“Least modification principle for case-based reasoning: a software project planning experience”. *Expert Systems with Applications*, vol. 30-2, 2006, pp 190-202.

[LEO94] LEON, V.J., WU, S.D., STORER, R.H.“Robustness measures and robust scheduling for job shops”. *IIE Transactions*,vol. 26-5, 1994,pp. 32– 41.

[LEU05] LEUS, R., HERROELEN,W. “The complexity of machine scheduling for stability with a single disrupted job”. *Operations Research Letters*, vol. 33-2, 2005, pp. 151–156.

[LEY00] LEYMANN, F., ROLLER, D.“Production workflow: concepts and techniques”. Upper Saddle River: Prentice Hall, 2000.

[LIA02] LIAO REN, CHEN QING-XIN, MAO NIN.“A heuristic algorithm for resource-constrained project scheduling”*Journal of Industrial Engineering and Engineering Management*, vol. 16, 2002,pp. 100-103.

[LOV01] LOVA A., TORMOS P."Analysis of scheduling schemes and heuristic rules performance in resourceconstrained multiple project scheduling". *Annals of Operations Research*, vol. 102(1-4), , 2001, pp. 263-286.

[LYC04] LYCETT, M., RASSAU, A., DANSON, J. "Programme management: a critical review". *International Journal of Project Management*, vol. 22-4, Maio 2004, pp. 289-299.

[MAC93] MACCARTHY, B. L., LIU, J.. "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling". *International Journal of Production Research,*vol. 31-1, 1993, pp. 59–79.

[MAC00] MACHADO, L.F.C. "Modelo para Definição de Processos de Software na Estação TABA". Dissertação de Mestrado, COPPE/UFRJ, 2000.

[MCC96] MC'CONNELL, S. "Rapid Development: Taming Wild Software Schedules",

Microsoft Press, 1996, 647p.

[MEH99] MEHTA, S. V., UZSOY, R. "Predictable scheduling of a single machine subject to breakdowns". *International Journal of Computer Integrated Manufacturing*, vol. 12-1, 1995, pp. 15–38.

[MEZ00] MEZIANE, F., VADERA, S., KOBBACY, K., PROUDLOVE, N. "Intelligent systems in manufacturing: current developments and future prospects". *Integrated Manufacturing Systems*, vol. 11-4, 2000, pp. 218–238.

[MIL94] MILES M.B., HUBERMAN A.M. "Qualitative Data Analysis". London: Sage Publications, 2^{nd} edition, 1994.

[MIL06] MILOSEVIC, D., PATANAKUL, P. "Gerentes de Multiprojetos – Quais competências são necessárias?", Revista Mundo PM, Número 11, 2006,pp. 62-68.

[MIY95] MIYASHITA, K., SYCARA, K. "CABINS: a framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair". *Artificial Intelligence*, vol. 76-1, 1995, pp. 377–426.

[MOU01] MOURA, A. V. "Especificações em Z: Uma Introdução". UNICAMP, 2001.

[MUH82] MUHLEMANN, A.P., LOCKETT, G., FARN, C.K. "Job shop scheduling heuristics and frequency of scheduling". *International Journal of Production Research*, vol. 20-2, 1982, pp. 227–241.

[MUR89] MURATA, T. "Petri Nets: Properties, Analysis and Applications". *Proceedings of the IEEE*, vol. 77-4, 1989.

[MYE07] MYERS, B."Foundations of WF: an introduction to Windows Workflow Foundation". Apress, 2007.

[OAS01] OASIS COMMITEE. "RELAX NG Specification". The Organization for the Advancement of Structured Information Standards, 2001. Capturado em: http://www.relaxng.org/spec-20011203.html, Maio, 2012.

[OAT06] OATES, B. “Researching Information Systems and Computing”.Sage Publications Ltda, 2006, 341p.

[ODO99] O’DONOVAN, R., UZSOY, R., MCKAY, K. N. “Predictable scheduling of a single machine with breakdowns and sensitive jobs”. *International Journal of Production Research*, vol. 37-18, 1999, pp. 4217–4233.

[OHA96] O’HARE, G., JENNINGS, N. “Foundations of distributed artificial intelligence”. New York: Wiley, 1996.

[OKA00] O’KANE, J. F. “A knowledge-based system for reactive scheduling decision-making in FMS”. *Journal of Intelligent Manufacturing,*vol. 11-5, 2000, pp. 461–474.

[OMG02] OBJECT MANAGEMENT GROUP. “Software & Systems Process Engineering Metamodel Specification 1.1”.Capturado em: http://www.omg.org/cgi-bin/doc?formal/02-11-14, Março 2012.

[OMG09] OBJECT MANAGEMENT GROUP. “Omg unified modeling language (omg uml) infrastructure v2.1.2”. Capturado em: http://www.omg.org/docs/formal/07-11-04.pdf, Agosto de 2009.

[OMG10] OBJECT MANAGEMENT GROUP. “Unified modeling language specification v1.5”. Capturado em: http://www.omg.org/docs/formal/03-03-01.pdf, Novembro 2010.

[OMG11] OBJECT MANAGEMENT GROUP. “Mof 2.0 Facility And Object Lifecycle”. Capturado em: http://www.omg.org/spec/SPEM/2.0/, Março 2011.

[OMG12] OBJECT MANAGEMENT GROUP. “Software & Systems Process Engineering Metamodel Specification 2.0”. Capturado em: http://www.omg.org/spec/SPEM/2.0/, Março 2012.

[OPE09] OPEN Consortium. “Open Web Site”.Capturado em:www.open.org.au, Agosto 2009.

[OUE03] OUELHADJ, D., COWLING, P. I., PETROVIC, S. “Utility and stability measures for agent-based dynamic scheduling of steel continuous casting”. In: Proceedings of the IEEE international conference on robotics and automation, Taipei, Taiwan, 2003, pp. 175–180.

[OVA94] OVACIK, I. M., UZSOY, R. “Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequencedependent set-up times”. *International Journal of Production Research*, vol. 32-6, 1994, pp. 1243–1263.

[PAR96] PARUNAK, H. V. “Applicat 1ions of distributed artificial Intelligence in industry”. In: G. M. P. O’Hare & N. R. Jennings (Eds.), Foundation of distributed artificial intelligence. New York: Wiley- Interscience. Chap. 4, 1996.

[PAR00] PARUNAK, H.V. “Agents in overalls: experiences and issues in the development and deployment of industrial agent-based systems”. *International Journal of Cooperative Information Systems*, vol. 9-3, 2000, 209–227.

[PAR11] PARK, S., BAE D.H. “An Approach to Analysing the Software Process Change Impact Using Process Slicing and Simulation”.*The Journal of Systems and Software*, 2011, pp. 528-543.

[PEP09] PROCESS ENGINEERING PROCESS. “A RUP Adoption Process”. Capturado em: http://www-128.ibm.com/developerworks/rational/library/6001.html, Agosto 2009.

[PER05] GONZALEZ-PEREZ, C., HENDERSON-SELLERS, B. “Templates and Resources in Software Development Methodologies”.*Journal of Object Technology*, 2005.

[PER07] GONZALEZ-PEREZ, C., HENDERSON-SELLERS, B. “Modelling Software Development Methodologies: A Conceptual Foundation”.*The Journal of Systems and Software*, 2007, pp. 1778-1796.

[PET81] PETERSON J.L. “Petri net theory and the modeling of systems”. New Jersey, USA: Prentice Hall, 1981, pp. 88-95.

[PFL09] PFLEEGER, S. L., ATLEE, J. “Software Engineering: Theory and Practice”. Prentice Hall, 4th edition, 2009.

[PLE98] PLEKHANOVA, V. “On Project Management Scheduling where Human Resource is a Critical Variable”. In: 6th European Workshop on Software Process Technology, 1998, pp. 116-121.

[PMI08] PROJECT MANAGEMENT INSTITUTE. “PMBOK - A Guide to the Project Management Body of Knowledge”. Newtown Square, PA: Project Management Institute, 4th edition, 2008.

[PRA04] PRADO, D. “Pert/CPM". Nova Lima: INDG, 2004, 173p.

[PRE09] PRESSMAN, R. “Software engineering: a practitioner's approach”. McGraw-Hill, 7th edition, 2009.

[RAT09] RATIONAL SOFTWARE CORPORATION. “Rational Unified Process: Best Practices for Software Development Teams”. Capturado em: http://www.rational.com/products/rup/whitepapers.jsp, Novembro 2009.

[REA05] REA, L.M., PARKER, R.“Designing and Conducting Survey Research: A Comprehensive Guide”.Jossey-Bass. 33rd edition, 2005.

[RED01] REDDY, J.P., KUMANAN, S., CHETTY, O.V.K. “Application of Petri nets and a genetic algorithm to multimode multi-resource constrained project scheduling”. *International Journal of Advanced Manufacturing Technology*, vol. 17-4, 2001,pp. 305-314.

[REH07] REHMAN, A., HUSSAIN, R. “Software Project Management Methodologies/Frameworks Dynamics ‘A Comparative Approach’”. In: 7th International Conference on Information and Emerging Technologies, Karachi, Pakistan, July 2007, pp. 1-5.

[RIH01] RIHA, A., PECHOUCEK, M., KRAUTWUNNOVA, H., CHARVA, P., KOUMPIS, A. “Adoption of an Agent-Based Production Planning Technology in the Manufacturing Industry”. In: 12th International Workshop on Database and Expert Systems Applications, pp. 640-646, 2001.

[ROS06] ROSITO, M. C.; CALLEGARI, D.; BASTOS, R. M. “Metamodelos de processos de desenvolvimento de software: Um estudo comparativo”. In: III Simpósio Brasileiro de Sistemas de Informação, Curitiba, 2006, 8p.

[ROS08a] ROSITO, M., CALLEGARI, D., BASTOS, R.“Gerência de Projetos e Processos de Desenvolvimento de Software: uma proposta de integração”. In: IV Simpósio Brasileiro de Sistemas de Informação, 2008, Rio de Janeiro.

[ROS08b] ROSITO, M.; CALLEGARI, D.; BASTOS, R. M. “Gerência de Projetos e Processos de Desenvolvimento de Software: uma proposta de integração”. iSys – Revista Brasileira de Sistemas de Informação - PPGI / UNIRIO, vol. 1, Outubro 2008, pp. o88-115.

[ROS12a] ROSITO, M., BASTOS, R. “A Systematic Review on the Integration of Project Management with Organizational Flows”. In: ICIS 2012 - International Conference on Information Systems, Penang, Malaysia, 2012.

[ROS12b] ROSITO, M., CALLEGARI, D., BASTOS, R. “Project Management and Software Development Processes: Integrating PMBOK and OPEN”. *International Journal of Computer and Communication Engineering*, vol. 6, 2012, pp. 139-147.

[ROS12c] ROSITO, M., BASTOS, R. “An Integrated Environment for Software Project Planning”. In: IADIS 2012 - International Conference Applied Computing, Madrid, Spain, 2012.

[ROS12d] ROSITO, M., BASTOS, R. “A Model to Integrate Software Project Management with Organizational Workflows”. In: ISDA 2012 - International Conference on Intelligent Systems Design and Applications, Kochi, India, 2012.

[ROS13] ROSITO, M., RIBEIRO, M., BASTOS, R. “An Experimental Study on the Dynamic Reconfiguration of Software Projects”. In: ICEIS 2013 - International Conference on Enterprise Information Systems , Angers, France, 2013.

[ROS14] ROSITO, M., BASTOS, R. “SPIM: An Integrated Model of Software Project Management and Organizational Workflows”. *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 6, 2014, pp. 160-179.

[SAB00] SABUNCUOGLU, I., & BAYIZ, M. “Analysis of reactive scheduling problems in a job shop environment”. *European Journal of Operational Research*, vol. 126-3, 2000, pp. 567–586.

[SAR90] SARIN, S. C., SALGAME, R. R. “Development of a knowledgebased system for dynamic scheduling”. *International Journal of Production Research*, vol. 28-8, 1990, pp. 1499–1513.

[SCH94] SCHMIDT, G. “How to apply fuzzy logic to reactive scheduling”. In: Knowledge-based reactive scheduling, Amsterdam: North-Holland, 1994, pp. 57–67.

[SCH01] SCHIMDT, R., LYYTINEM, K., KEIL, M., CULE, P. “Identifying software project risks: an international delphi study”. *Journal of Management Information Systems*, vol.17-4, 2001, pp. 5-36.

[SCH02] SCHWALBE K. “Information Technology Project Management”. Thomson Learning, 2nd edition, 2002.

[SCH10] SCHLÖSSER, R. “Gerenciamento Distribuído de Agendas de Recursos para Projetos de Desenvolvimento de Software baseado em Sistemas Multiagentes”. Dissertação de Mestrado, PUC-RS Porto Alegre, 2010.

[SGI10] STANDISH GROUP INTERNATIONAL. “Chaos: A Recipe for Sucess”. Artigo desenvolvido pelo Standish Group International Inc., Capturado em: http://www.standishgroup.com/sample_research/, Junho 2010.

[SHE99] SHEN, W., NORRIE, D. H. “Agent based systems for intelligent manufacturing: a state of the art survey”. *International Journal of Knowledge and Information* Systems, vol. 1-2, 1999, pp. 129–156.

[SHE01] SHEN, W., NORRIE, D. H., BARTHES, J. P. A. “Multi-agent systems for concurrent intelligent design and manufacturing”. London: Taylor & Francis, 2001.

[SHU96] SHUKLA, C. S., CHEN, F. F. “The state of the art in intelligent real-time FMS control: a comprehensive survey”. *Journal of Intelligent Manufacturing*, vol. 7, 1996,pp. 441–455.

[SMI80] SMITH, R. “The Contract Net Protocol: High-Level Communication andControl in a Distributed Problem Solver”. *IEEE Transactions on Computer*, vol. C-29, N. 12, 1980, pp. 1104-1113.

[SMU95] SMULLYAN, R.M. “First-Order Logic”, Dover Publications, Inc. New York, 1995, 157p.

[SOD08] SÖDERHOLM, A. “Project Management of Unexpected Events”. *International Journal of Project Management,* vol. 26, 2008, pp. 80-86.

[SOL99] SOLINGEN, R., BERGHOUT, E. “The Goal/Question/Metric Method”. New York: McGraw-Hill, 1999.

[SOM06] SOMMERVILLE, I. “Software engineering”. Addison-Wesley, 8th edition, 2006.

[STO96] STOOP, P.P.M., WEIRS, V.C.S. “The complexity of scheduling in practice”.*International Journal of Operations and Production management*, vol. 16-10, 1996, pp. 37–53.

[SUN01] SUN, J., XUE, D. “A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources”. *Computers in Industry*,vol. 46-2, 2001, pp. 189–207.

[SUR93] SURESH, V., CHAUDHURI, D.“Dynamilc scheduling a survey of research”. *International Journal of Production Economics*, vol. 32-1, 1993, pp. 53–63.

[SZE94] SZELKE, E., KERR, R. M. “Knowledge-based reactive scheduling”. Amsterdam: North-Holland, 1994.

[THA97] THARUMARAJAH, A., BEMELMAN, R. “Approaches and issues in scheduling a distributed shop-floor environment”. *Computers in Industry*, vol. 34-1, 1997, pp. 95–109.

[TRA02] TRAVASSOS, G.H., GUROV, D., AMARAL, G. “Introdução a Engenharia deSoftware Experimental”. Relatório Técnico, PESC 590/02, COPE/UFRJ,2002, 66 p.

[TSA03] TSAI, H., MOSKOWITZ H., LEE, L. “Human resource selection for software development projects using taguchi’s parameter design”. *European Journal of Operational Research*, vol. 151-1, 2003, pp. 167-180.

[TYR00] TYRRELL S. “The Many Dimensions of the Software Process”.*Crossroads*, vol. 6-4, 2000, pp.22-26.

[UHE03] UHER, T. “Programming And Scheduling Techniques”. University of New South Wales Press: Sydney, 2003, 295p.

[VAN98] VAN DER AALST W. ”The Application of Petri Nets to Workflow Management”. *The Journal of Circuits, Systems and Computers*, vol. 8-1, 1998, pp. 21-66.

[VER05] VERNER, M., CERPA, N. “Australian software development: what software project management practices leed to success?”. In: IEEE Proceedings of ASWEC, Ed Paul Strooper, Brisbane, Australia, 2005, pp. 70-77.

[VIE00a] VIEIRA, G.E., HERRMANN, J.W., LIN, E. “Analytical models to predict the performance of a single machine system under periodic and event-driven rescheduling strategies”. *International Journal of Production Research*, vol. 38-8, 2000, pp. 1899–1915.

[VIE00b] VIEIRA, G.E., HERMANN, J.W., LIN, E. “Predicting the performance of rescheduling strategies for parallel machine systems”. *Journal of Manufacturing Systems*, vol. 19-4, 2000, pp. 256–266.

[VIE03] VIEIRA, G.E., HERMANN, J.W., LIN, E. “Rescheduling manufacturing systems: a framework of strategies, policies and methods”. *Journal of Scheduling*, vol. 6-1, 2003, pp. 36–92.

[WAR03] WARMER, J., KLEPPE, A. “The Object Constraint Language Second Edition – Getting Your Models Ready for MDA”, 2003.

[WCG04] W3C XML WORKING GROUP. Extensible Markup Language (XML) 1.1, W3C Recommendation 04 February 2004. Capturado em:

http://www.w3.org/TR/2004/REC-xml11-20040204/, Maio, 2012

.

[WEB02] WEBER, M., KINDLER, E. “The Petri Net Markup Language”. In: Petri Net Technology for Communication Based Systems, 2002, vol. 2472, p. 124-144.

[WER96] WERNER, C.M.L., TRAVASSOS, G.H., ROCHA, A.R.C., WERNECK, V.M. “Memphis: Um Ambiente para Desenvolvimento de Software Baseado em Reutilização.” Relatório Técnico, COPPE/UFRJ, 1996.

[WOH00] WOHLIN, C., RUNESON, P., HÖST, M., OHLSSON, M., REGNELL, B., WESSLÉN, A. “Experimentation in software engineering: an introduction”. Boston: Kluwer Academic Publishers, 204 p, 2000.

[WUS91] WU, S. D., STORER, R. H., CHANG, P. C. “A rescheduling procedure for manufacturing systems under random disruptions”. In: Proceedings joint USA/German conference on new directions for operations research in manufacturing, 1991, pp. 292–306.

[WUS93] WU, S. D., STORER, R. H., CHANG, P. C. “One machine rescheduling heuristics with efficiency and stability as criteria”. *Computers Operations Research,* vol. 20-1, 1993, pp. 1–14.

[XU05] XU, P., RAMESH, B. “Knowledge Support in Software Process Tailoring”. In: Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS), 2005.

[YAS11] YASPER. “Yasper User Guide”. Capturado em: http://www.yasper.org/, Setembro 2011.

[YOO01] YOON, I., MIN, S., BAE, D. “Tailoring and Verifying Software Process”.In: Institute of Electrical and Electronic Engineers - IEEE,2001, pp.202-209.

[YOU01] YOUSSEF, H., SAIT, S. M., ADICHE, H. “Evolutionary algorithms, simulated annealing and tabu search: a comparative study”. *Engineering Applications of Artificial Intelligence*, vol. 14-2, 2001, pp. 167–181.

[YUA05] YUAN CHONG-YI. “Petri net theory and its application”. Beijing: Publishing House of Electronics Industry, 2005: 135-145.

[ZHO95] ZHOU, M.C., ZURAWSKI, R. "Introduction to Petri Nets in Flexible and Agile Automation".Kluwer Academic Publishers, Boston, MA, 1-42, 1995.

[ZHO01] ZHOU, H., FENG, Y., HAN, L. “The hybrid heuristic genetic algorithm for job shop scheduling”. *Computers and Industrial Engineering*, vol. 40-3, 2001, pp. 191–200.

[ZWE94] ZWEBEN, M., FOX, M. S.“Intelligent scheduling”. San Mateo: Morgan Kaufmann, 1994.

APÊNDICE A – DESCRIÇÃO DAS PRINCIPAIS CLASSES DO SPEM 2.0

Este apêndice apresenta o funcionamento e a descrição das principais classes dos pacotes do SPEM 2.0.

A.1 PACOTE CORE

O pacote `Core` é o pacote núcleo do SPEM 2.0. Nesse pacote, as metaclasses e abstrações que constroem a base para todos os outros pacotes do metamodelo são definidas.

Basicamente, o pacote `Core` define duas capacidades para o SPEM 2.0:

- A habilidade para usuários criarem qualificações para diferentes metaclasses do SPEM 2.0 distinguindo elas com diferentes “`kinds`”.
- Um conjunto de metaclasses abstratas para expressar o trabalho nos processos de software. Todas as metaclasses do metamodelo SPEM 2.0 que derivam das metaclasses de trabalho definidas no `Core` tem como objetivo mapear os modelos de comportamento do processo.

As metaclasses definidas no pacote `Core` tem como super classes as metaclasses definidas na UML 2.0 e serão exibidas nas Figura 73e Figura 74.

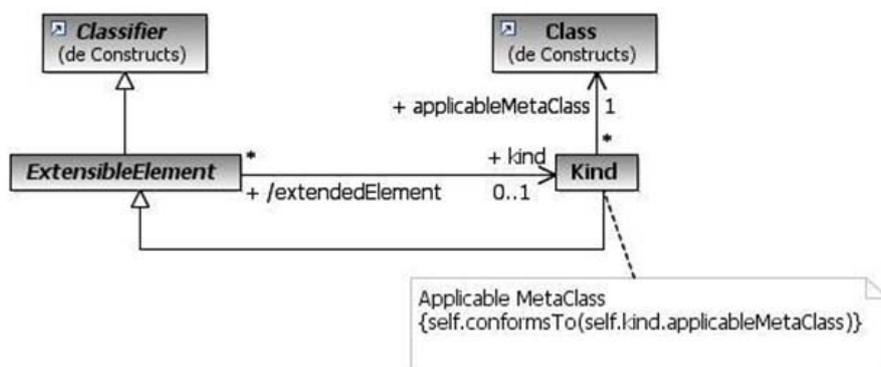



Figura 73. Metaclasses ExtensibleElement e Kind definidas no pacote Core

A metaclasse `ExtensibleElement`, vista na Figura 73, é uma generalização abstrata que representa qualquer metaclasse do metamodelo SPEM 2.0 que pode ter uma qualificação definida pelo usuário. Cada metaclasse do metamodelo SPEM 2.0 que permite utilizar esta qualificação deriva direta ou indiretamente da metaclasse

`ExtensibleElement`. A metaclasse `kind` é uma especialização da metaclasse `ExtensibleElement`. Esta metaclasse é usada para qualificar outras instâncias de metaclasse do metamodelo SPEM 2.0 com tipos definidos pelo usuário.

Prosseguindo com e explanação sobre as metaclasses do pacote *Core*, têm-se as mostradas na Figura 74. Inicialmente, a metaclasse `ParameterDirectionKind` é uma enumeração que representa parâmetros de entrada (`in`), saída (`out`) e também de entrada e saída (`inout`) para as instâncias das metaclasses e subclasses de `WorkDefinition`. Esses parâmetros são definidos através do atributo direction definido para a classe `WorkDefinitionParameter` que é uma generalização abstrata para elementos do processo e representa os parâmetros para as metaclasses e subclasses de `WorkDefinition`.

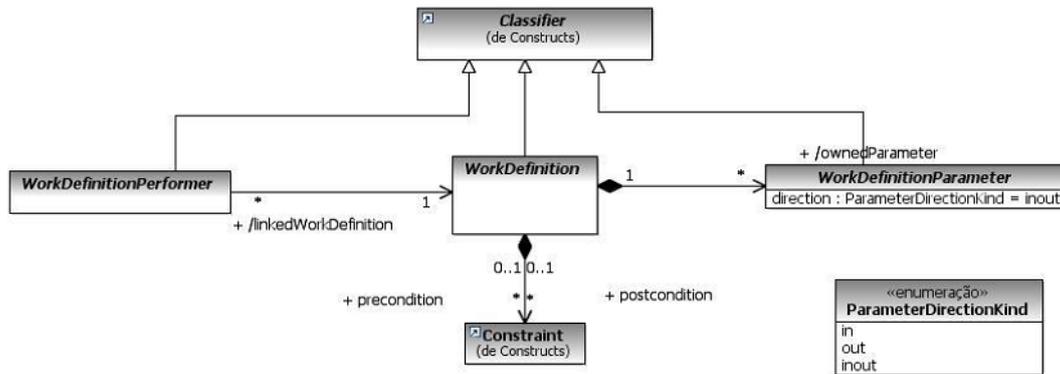



Figura 74. Metaclasses do pacote `Core`

A metaclasse `WorkDefinition` é uma metaclasse abstrata que generaliza todas as definições de trabalho no metamodelo SPEM 2.0. Nos pacotes `ProcessStructure` e `MethodContent`, respectivamente, essa metaclasse é especializada em atividades e tarefas. No pacote `Core`, a metaclasse `WorkDefinition` pode conter pré e pós condições que são representadas pela metaclasse Constraint da UML 2.0. A metaclasse `WorkDefinition` também pode possuir 0 ou vários parâmetros representados pela metaclasse `ParameterDirectionKind` e também pode ser associada a 0 ou várias metaclasses `WorkDefinitionPerformer`. A metaclasse `WorkDefinitionPerformer` é uma metaclasse abstrata que representa o relacionamento de um executor de trabalho para uma metaclasse `WorkDefinition`. Essa metaclasse será especializada em outros pacotes do metamodelo SPEM 2.0 em diferentes tipos de relacionamentos. Ela irá ser especializada, por exemplo, no pacote `ProcessStructure` em `ProcessPerformer`.

A.2 PACOTE PROCESS STRUCTURE

O pacote `ProcessStructure` é o pacote que define a base para todos os processos de software no metamodelo SPEM 2.0. Nesse pacote, processos são representados por uma estrutura de *Work* Brea*k*down *Element* – WBS que permite o aninhamento de atividades dentro de outras atividades ou, ainda, o aninhamento de outros elementos dentro de uma atividade. Desse modo, os elementos que podem ser aninhados em atividades são papéis, produtos de trabalho e também várias metaclasses que representam relacionamentos. A Figura 75 e a Figura 76 exibem, respectivamente, a taxonomia de metaclasses do pacote `ProcessStructure` e as principais associações definidas nele.

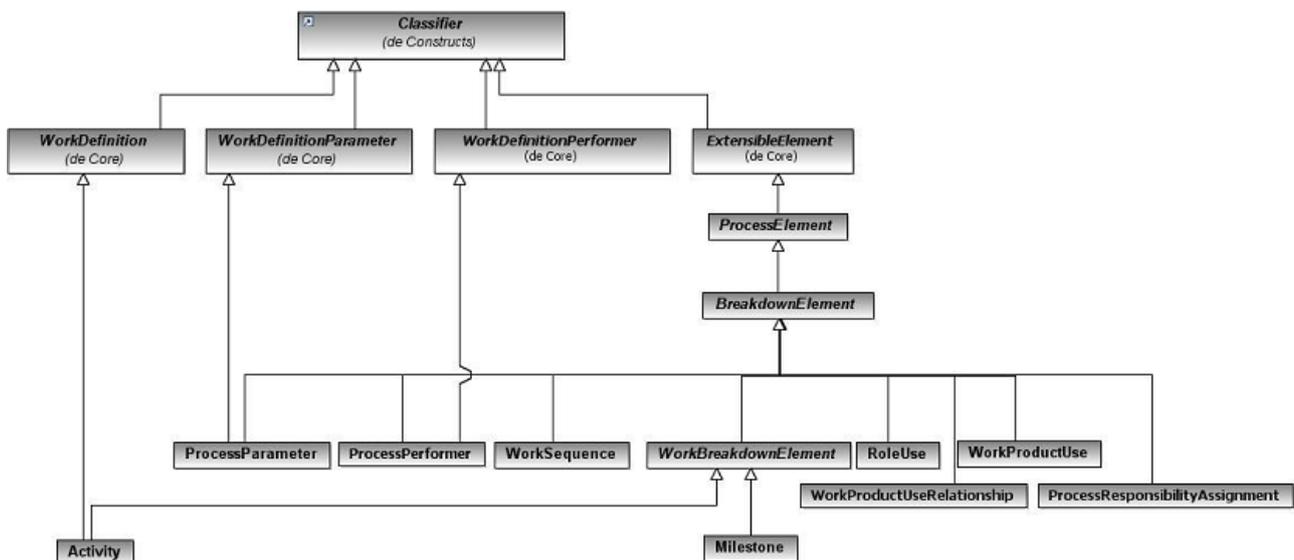



Figura 75. Taxonomida das metaclasses do pacote `ProcessStructure`

Na Figura 75 as metaclasses `Activity`, `ProcessPerformer`, `ProcessParameter` e `BreakdownElement` especializam, respectivamente, as metaclasses `WorkDefinition`, `WorkDefinitionPerformer`, `WorkDefinitionParameter` e `ExtensibleElement` do pacote `Core`.

As metaclasses abstratas `BreakdownElement` e `WorkBreakdownElement` permitem a representação de um processo através de uma estrutura WBS. A ideia é que a metaclasse `Activity` seja utilizada para instanciar os elementos que representam unidades de trabalho nos processos tais como as atividades e também seja utilizada para instanciar os elementos do processo que organizam as unidades de trabalho em definições de tempo tais como as fases, as iterações e o próprio processo. Já os outros elementos deste pacote que são `RoleUse`, `WorkProductUse`,

WorkProductUseRelationship, WorkSequence, Milestone, ProcessPerformer e ProcessResponsabilityAssignment podem ser instanciados dentro das atividades para representar as outras informações do processo de software.

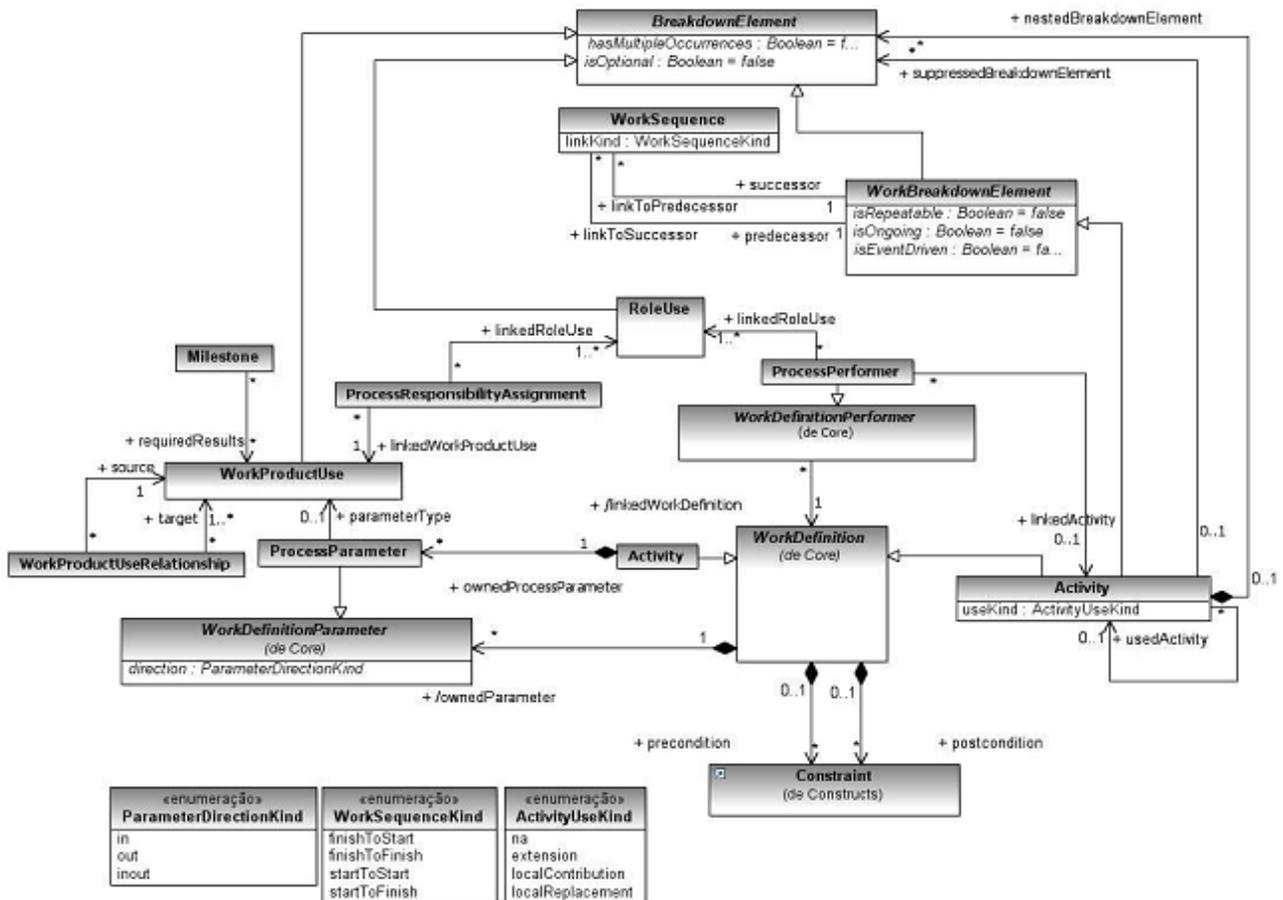



Figura 76. Metaclasses e associações do pacote ProcessStructure

As metaclasses Activity, RoleUse, WorkProductUse e Milestone representam elementos do processo de software. Já as metaclasses WorkProductUseRelationship, ProcessPerformer, ProcessResponsabilityAssignment, ProcessParameter e WorkSequence representam os relacionamentos entre esses elementos. WorkProductUseRelationship estabelece relações entre WorkProductUses; ProcessPerformer estabelece a relação entre as atividades e os papéis no processo de software; ProcessResponsabilityAssignment estabelece a relação de responsabilidade entre os papéis e os produtos de trabalho; ProcessParameter estabelece através do atributo direction e da metaclasse de enumeração ParameterDirectionKind os parâmetros de entrada e/ou saída para as atividades em termos de produtos de trabalho.

A metaclasse WorkSequence permite estabelecer sequenciamento entre as atividades utilizando o atributo linkKind e a metaclasse de enumeração WorkSequenceKind. As relações de sequenciamento permitem estabelecer relações de dependência entre as atividades em que uma atividade depende do início ou fim de outra(s) atividade(s) para poder iniciar ou terminar.

Por fim, existem o auto relacionamentousedActivity da metaclasse Activity e as relações suppressedBreakdownElement e nestedBreakdownElement entre as metaclasses Activity e BreakdownElement. A relação nestedBreakdownElement permite o aninhamento de elementos dentro de uma atividade. Já as relações usedActivity e suppressedBreakdownElement permitem, respectivamente, um mecanismo de reuso e supressão de elementos do processo.

A.3 PACOTE MANAGED CONTENT

O pacote ManagedContent define os conceitos fundamentais para gerenciar as descrições textuais para os processos (elementos do ProcessStructure) e os elementos do MethodContent no SPEM 2.0. Esse pacote introduz a metaclasse abstrata DescribableElement que, através do mecanismo de merge, serve como uma super classe para elementos de processo definidos no pacote ProcessStructure e também para as metaclasses do pacote MethodContent. O elemento DescribableElement é composto de uma metaclasse ContentDescription que permite relacionar descrições textuais para os elementos. A Figura 77 mostra as metaclasses e relações do pacote ManagedContent.

Na Figura 77, é possível verificar que a metaclasse DescribableElement é uma especialização da metaclasse ExtensibleElement do pacote Core. A metaclasse DescribableElement possui uma relação de composição com a metaclasse ContentDescription que possui vários atributos relacionados com descrição textual, bem como relação de composição com uma metaclasse denominada *Section*. Isso é o que permite que todas as especializações da metaclasse DescribableElement possuam descrições textuais e também que se utilizem do conceito de Seção. Observa-se também que a metaclasse ProcessElement (definida no pacote ProcessStrucure) aparece como uma especialização da metaclasse DescribableElement. Através dessa relação é que os elementos do pacote ProcessStructure podem possuir descrições textuais.

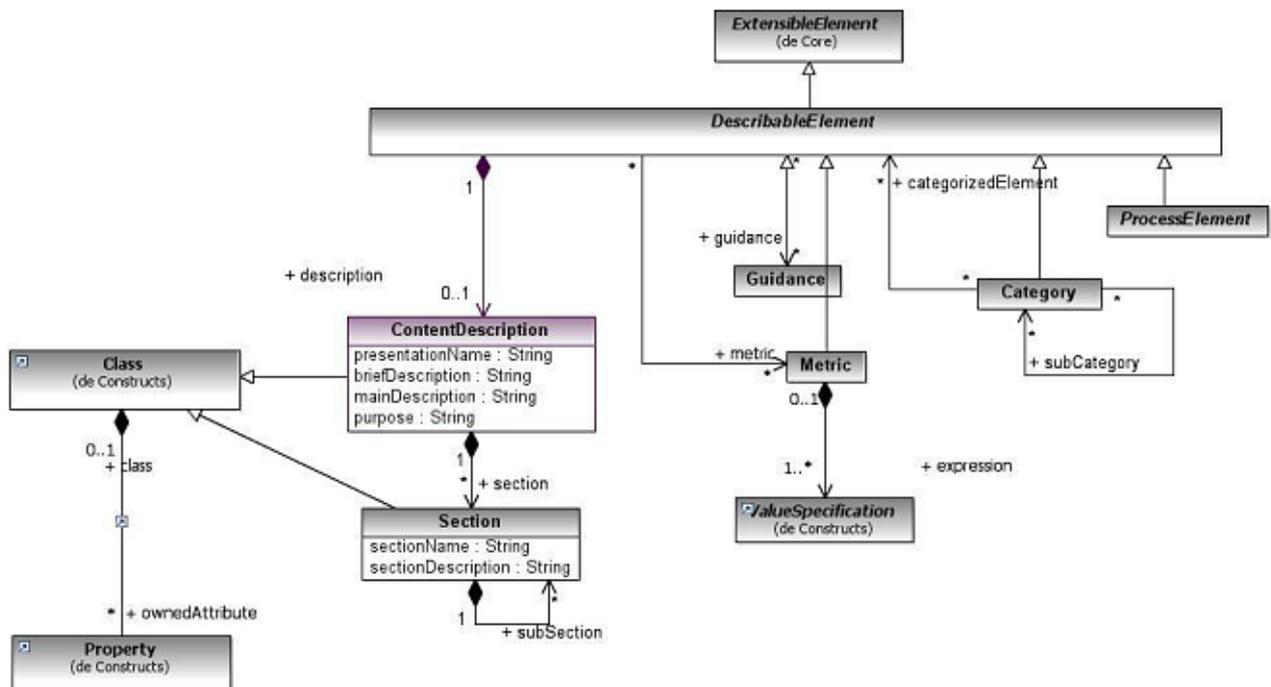



Figura 77. Metaclasses e associações do pacote `ManagedContent`

Existem também as metaclasses `Guidance`, `Category` e `Metric` que especializam a metaclasse `DescribableElement`. A metaclasse `Guidance`, além de ser uma especialização de `DescribableElement`, possui relação com essa metaclasse. Isso permite que guias contendo informação adicional possam ser associados a qualquer elemento `DescribableElement`. As metaclasses `Category` e `Metric`, igualmente a metaclasse *Guidance*, também possuem relação com a metaclasse `DescribableElement`. Tais metaclasses podem ser usadas, respectivamente, para categorizar os elementos `DescribableElement` e para associar uma ou mais restrições que fornecem medidas para qualquer elemento `DescribableElement`.

A.4 PACOTE PROCESS BEHAVIOR

O metamodelo SPEM 2.0 não fornece conceitos para modelagem da execução de processos de software. De acordo com a especificação desse metamodelo, ele apenas define a habilidade para que implementadores escolham uma abordagem de modelagem de comportamento que melhor atendam suas necessidades. Partindo do contexto que esse pacote não é detalhado no metamodelo SPEM 2.0 e que esta pesquisa não considera aspectos de execução de processos, o pacote `ProcessBehavior` não é descrito neste trabalho em termos de suas classes e relacionamentos.

A.5 PACOTE METHOD CONTENT

O pacote `MethodContent` define os elementos que formam uma base de conteúdo para ser utilizada em qualquer processo de software. Os principais elementos desse pacote são Papéis, Tarefas e Produtos de Trabalho. AFigura 78 e a Figura 79exibem, respectivamente, a taxonomia dos elementos do pacote `MethodContent` e como as principais metaclasses desse pacote estão relacionadas.

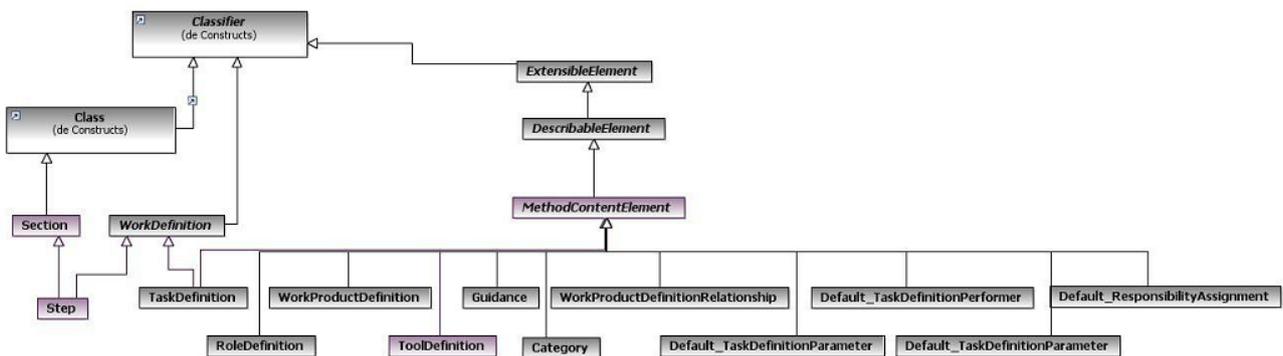



Figura 78. Taxonomida das metaclasses do pacote `MethodContent`

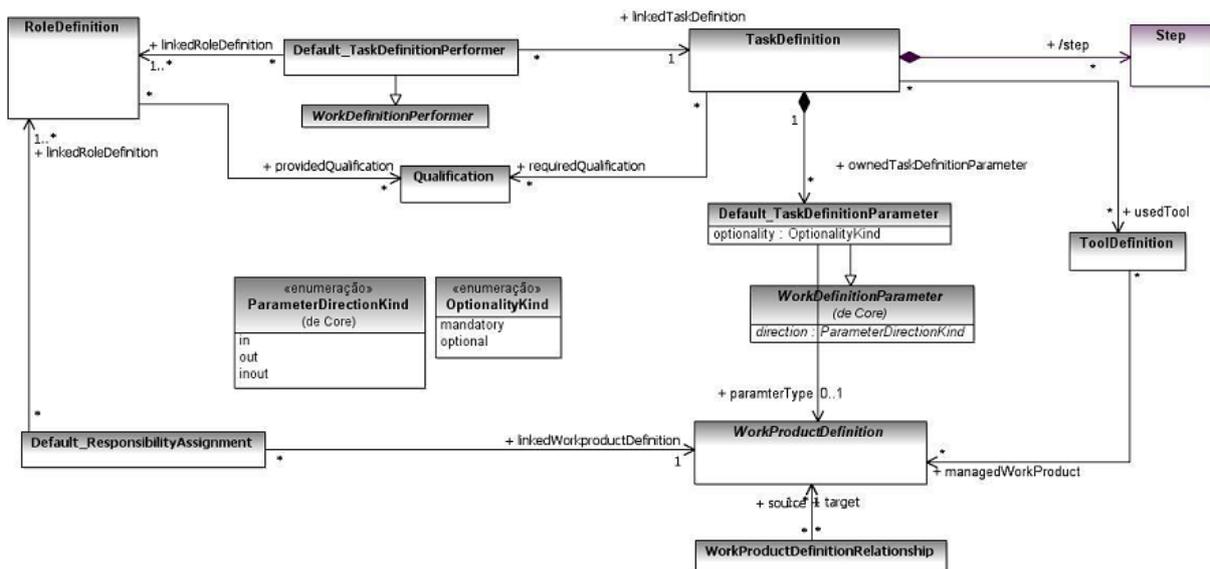



Figura 79. Metaclasses e associações do pacote `MethodContent`

Na Figura 78, observa-se que todos os elementos definidos nesse pacote são especializações da metaclasse `MethodContentElement` que, por sua vez, especializa a metaclasse `DescribableElement`. Os elementos introduzidos pelo mecanismo de merge neste pacote são `Step`, *TaskDefinition*, `RoleDefinition`, `WorkProductDefinition`, `ToolDefinition`, `Qualification`, `WorkProductDefinitionRelationship`, `Default_TaskDefinitionPerformer`,

Default_TaskDefinitionParameter e Default_ResponsabilityAssignment. Já a Figura 79 exibe como os elementos do pacote MethodContent se relacionam. Basicamente, a metaclasse TaskDefinition pode ser considerada como elemento central do pacote. Essa metaclasse possui relação com a metaclasse RoleDefinition através das metaclasses Default_TaskDefinitionPerformer e Qualification; relação com a metaclasse WorkProductDefinition através da metaclasse Default_TaskDefinitionParameter; relação com a metaclasse ToolDefinition e relação de composição com a metaclasse Step. Os outros relacionamentos desse pacote são o auto-relacionamento para a metaclasse WorkProductDefinition representado pela metaclasse WorkProductDefinitionRelationship e o relacionamento entre a metaclasse WorkProductDefinition e RoleDefinition estabelecido pela metaclasse Default_ResponsabilityAssignment.

A.6 PACOTE PROCESS WITH METHODS

O pacote Process with Methods liga os conceitos definidos no pacote ProcessStructure com os conceitos definidos no pacote Method Content. Assim, é possível que processos de software sejam montados, utilizando os conceitos pré-definidos no Method Content. A Figura 80 mostra a taxonomia das metaclasses do pacote Process with Methods. Nessa Figura, é possível observar que as novas metaclasses incluídas neste pacote são TeamProfile, CompositeRole, MethodContentUse e TaskUse. Além disso, as metaclasses RoleUse e WorkProductUse que já haviam sido definidas no pacote ProcessStructure são, agora, especializações da nova metaclasse MethodContentUse.

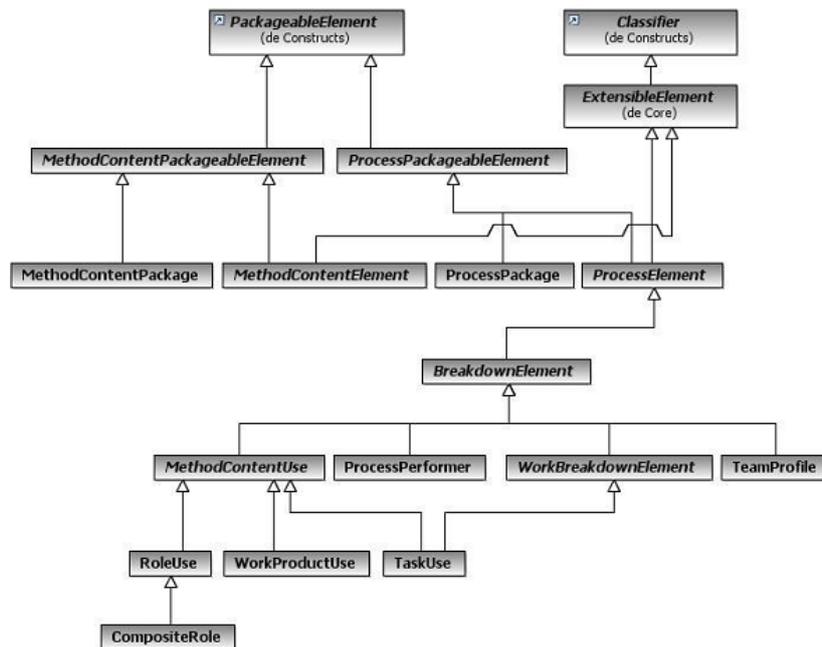



Figura 80. Taxonomida das metaclasses do pacote `Process with Methods`

As principais relações das metaclasses no pacote `Process with Methods` são mostradas na Figura 81. Nesta Figura, novamente percebe-se que o elemento tarefa é o elemento central. A metaclasse `TaskUse` assume praticamente as mesmas relações daquelas descritas no pacote `MethodContent`. A grande diferença nesse pacote é que a nova metaclasse `TaskUse` assume relações com elementos definidos no pacote `ProcessStructure`. Além disso, relacionamentos novos são estabelecidos no pacote `Process with Methods`. Esses relacionamentos ligam os elementos `TaskUse`, `WorkProductUse` e `RoleUse`, respectivamente, com os elementos `TaskDefinition`, `WorkProductDefinition` e `RoleDefinition` do pacote `Method Content`.

Através das relações acima entre as metaclasses do tipo `Use` e `Definition` é que um processo reutiliza os elementos definidos no `MethodContent`. A idéia do metamodelo SPEM 2.0 é que um repositório de conteúdo seja montado através do pacote `Method Content` e que as estruturas dos processos de software (tal como o fluxo de atividades e tarefas) sejam estabelecidas com os conceitos definidos no `ProcessStructure`. A junção desses conceitos é realizada justamente no pacote `Process withMethods`, o qual, através do mecanismo de merge, possui ligações com ambos os pacotes `Method Content` e `Process Structure` (conforme Figura 5).

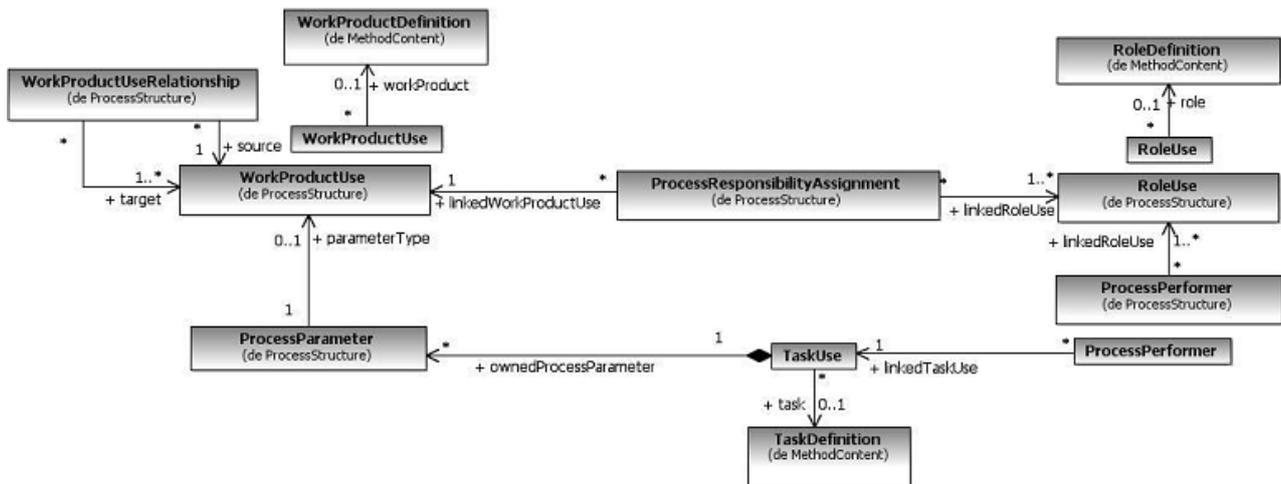



Figura 81. Metaclasses e associações do pacote `Process withMethods`

A.7 PACOTE METHOD PLUGIN

O pacote `Method Plugin` define a capacidade para gerenciar bibliotecas de `Method Content` e `Process`. Esse pacote endereça o interesse de estabelecer grandes bibliotecas por definir `Method Plugins` and `Method Configurations`. Adicionalmente, o pacote `Method Plugin` define mecanismos de variabilidade e extensibilidade para o `Method Content` e `Process`. Tais mecanismos são considerados pelo metamodelo SPEM 2.0 como mecanismos de adaptação, os quais permitem que processos sejam criados sobre demanda ou adaptados de acordo com o contexto dos projetos de software.

As novas metaclasses incluídas no pacote `Method Plugin` são mostradas na Figura 82 que exibe a taxonomia de metaclasses para esse pacote. Observa-se ainda que novas metaclasses são incluídas nesse metamodelo e também que há novas definições de especialização. As principais novas metaclasses são as metaclasses `MethodConfiguration`, `MethodPlugin`, `MethodLibrary` e `VariabilityElement` e estão envolvidas na definição de bibliotecas e também com o mecanismo de adaptação proposto nesse pacote.

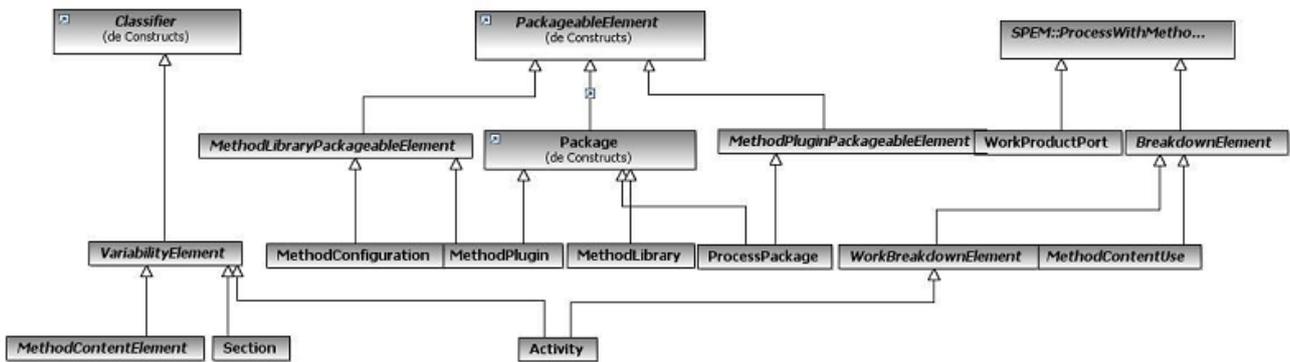



Figura 82. Taxonomida das metaclasses do pacote `MethodPlugin`

As relações entre as metaclasses `MethodConfiguration`, `MethodPlugin` e `MethodLibrary` podem ser vistas no diagrama de classes exibido na Figura 83. Nessa Figura, observa-se que uma biblioteca representada pela metaclasse `MethodLibrary` é uma composição da metaclasse `MethodPlugin` e da metaclasse `MethodConfiguration`.

Um *MethodPlugin* reprenta um container físico para `Content` (definido com metaclasses do pacote *Method*`Content`) e `ProcessPackages` (definido com metaclasses do pacote `Process Structure`, usando ou não as metaclasses do pacote `Method Content`). Na Figura 19, a metaclasse `MethodPlugin` é uma composição das metaclasses *ProcessPackage* e `MethodContentPackage`, as quais, respectivamente, guardam as metaclasses do pacote `Process Structure` e do pacote `Method Content`.

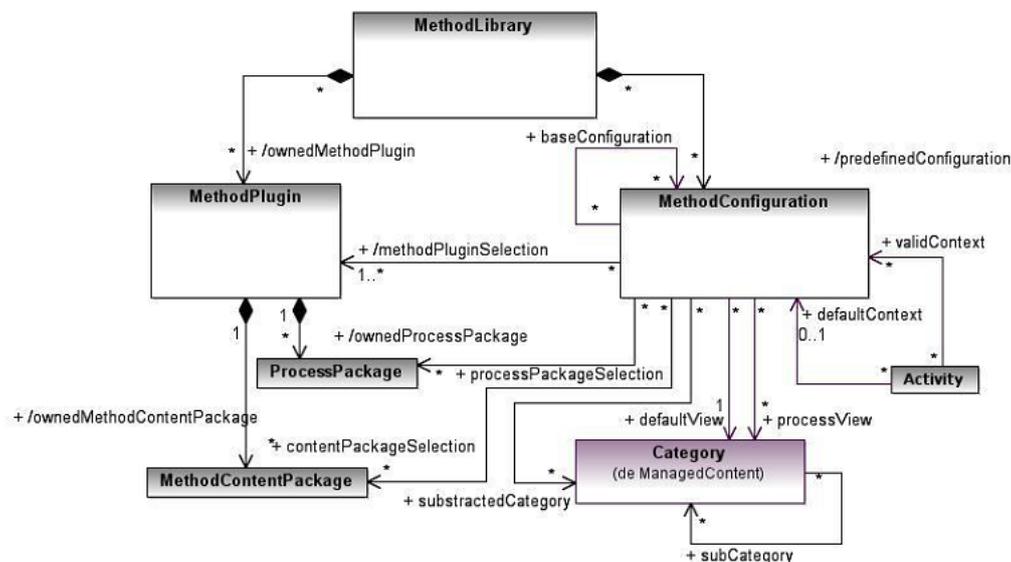



Figura 83. Associações entre as metaclasses `Method Library`, `Method Plugin` e `Method Configuration`

Um Method Configuration é uma coleção de Method Plugins selecionada e seus subconjuntos de MethodContentPackages e ProcessPackages. Esta metaclasse, MethodConfiguration, é utilizada no SPEM 2.0 para definir subconjuntos lógicos dentro de uma biblioteca (MethodLibrary).

A Figura 84 exibe a definição da metaclasse VariabilityElement no pacote MethodPlugin. Essa metaclasse e suas relações são considerados como mecanismos de adaptação do metamodelo SPEM 2.0.

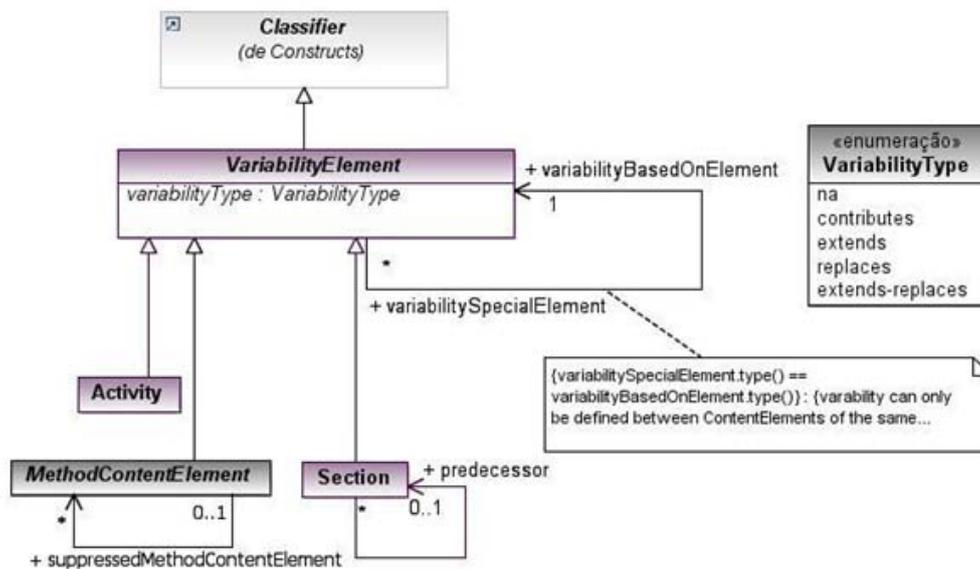



Figura 84. Metaclasse Variability definida no pacote MethodPlugin

APÊNDICE B – PROTOCOLO DA REVISÃO SISTEMÁTICA

Este apêndice contém a descrição do protocolo da revisão sistemática desta pesquisa.

B.1 QUESTÃO-FOCO

O objetivo geral desta revisão é encontrar pesquisas interessadas em propor soluções relacionadas com a reconfiguração dinâmica das redes de planejamento de projetos de software, considerando o planejamento e o replanejamento de suas atividades, tendo em vista a integração destes projetos com os fluxos organizacionais das empresas.

B.1.1 Qualidade e Amplitude da Questão

Esta seção objetiva descrever a sintaxe da questão de pesquisa – através dos itens problema e questão - e a sua semântica – através dos itens intervenção, controle, efeito, medição dos resultados, população, aplicação e projeto experimental.

- Problema: Durante o planejamento e execução de projetos de software, o gerente de projetos pode necessitar interagir com outros departamentos da organização a fim de obter informações relevantes para o projeto. Neste instante, há uma dissociação entre o fluxo de atividades de um projeto de software e os demais fluxos de atividades de suporte ao projeto da organização. Consequentemente, observa-se a necessidade de identificar os requisitos para uma solução de reconfiguração dinâmica de projetos de software que envolva múltiplos projetos simultâneos e a integração com os demais fluxos organizacionais, considerando o sequenciamento e a relação de dependência entre as atividades pertencentes a estes dois tipos de fluxos de trabalho.

- Questão de Pesquisa: Quais abordagens existentes na literatura permitem a reconfiguração dinâmica das redes de planejamento de projetos de software, que envolva múltiplos projetos simultâneos e a integração com os demais fluxos organizacionais?

Tabela 21: Palavras-chaves e sinônimos

Palavras-Chaves	Tradução em português	Sinônimos em inglês
Project Management	*Gerenciamento de projetos*	-
Business Process	*Processo de Negócio*	*Business Workflow*

Managerial Activity	*Atividade Gerencial*	*Enterprise activity* *Organizational Activity*
Productive Activity	*Atividade Produtiva*	-
Dynamic Reconfiguration	*Reconfiguração Dinâmica*	-

- Intervenção: Identificação e avaliação das soluções expostas nas pesquisas.

- Controle: inexistente.

- Efeito: definir o estado da arte sobre aspectos relacionados com a reconfiguração dinâmica de projetos de desenvolvimento de software.

- Medição de resultados: número de trabalhos identificados considerando os critérios de inclusão e exclusão de trabalhos.

- **População:** artigos publicados em periódicos e anais de conferências relacionados ao tema de pesquisa, cuja data de publicação seja posterior a 2000.

- Aplicação: pesquisadores da área e engenheiros de processo.

- Projeto experimental: não se aplica.

B.2 Seleção das Fontes

B.2.1 Definição dos Critérios de Seleção das Fontes

Disponibilidade de consulta a artigos completos (*full papers*) através da Web pelo convênio PUCRS-CAPES; existência de mecanismos de busca com suporte a inserção de operadores booleanos (como *and* e *or*); e base de dados atualizada (com publicações dos últimos 10 anos).

B.2.2 Idiomas

Inglês. A justificativa para esta escolha se deve à universalidade do idioma, sendo o padrão de conferências e periódicos internacionais.

B.2.3 Identificação das Fontes

- Método de pesquisa das fontes: através dos mecanismos de busca próprios do IEEE Xplore, ACM Digital Library, SpringerLink e Science@Direct.

- String de busca:*(project management OR scheduling) AND (business process OR workflow) AND (managerial activity OR enterprise activity OR organizational activity OR productive activity) AND (year >= 2000 AND year <= 2011)*

- Lista de fontes: vide Tabela 6do Capítulo 4.

B.2.4 Seleção das Fontes após Avaliação

Todas as fontes foram aprovadas pelo doutorando e pelo seu orientador.

B.2.5 Verificação de Referências

Todas as fontes selecionadas foram aprovadas.

B.1.3 SELEÇÃO DOS ESTUDOS

B.1.3.1 Definição dos Estudos

- Critérios de inclusão e exclusão de estudos: foram desconsiderados os estudos:

- C1. Que não estavam no formato de artigo completo (*full paper*).
- C2. Cujo conteúdo não estava relacionado a área de reconfiguração de projetos.
- C3. Trabalhos que citem importância da reconfiguração de projetos, mas que não possuam nenhum tipo de solução para isto.

- Definição do tipo de estudo: foram removidos estudos do tipo *roadmaps* (trabalhos que indicam os desafios e as direções a serem tomadas em uma área de pesquisa).

- Procedimentos para a seleção de estudos: execução da string de busca nos mecanismos de busca oferecidos em cada uma das fontes selecionadas. A análise da seleção dos estudos foi realizada em 3 passos: (1) leitura de títulos e resumos (*abstract*) dos artigos retornados; (2) leitura dos artigos completos; e (3) seleção dos artigos que traziam soluções relacionadas com a reconfiguração de projetos. Nessa última etapa alguns artigos que ficaram de fora desse universo foram utilizados como referencial teórico por apresentarem importantes conceituações.

B.3.2 Execução da Seleção

- Seleção inicial de estudos: a busca em todos os mecanismos resultou em 132 artigos.

- Avaliação da qualidade dos estudos: 10 estudos (ou 7,57% da amostra inicial) foram selecionados para a extração de informações.

- Revisão da seleção: a seleção de estudos foi aprovada.

B.1.4 EXTRAÇÃO DE INFORMAÇÕES

B.4.1 Critérios de Inclusão e Exclusão de Informações

Conforme descrito no Capítulo 4.

B.4.2 Formulários para Extração dos Dados

Os diferentes aspectos identificados estão descritos no Capítulo 4.

B.4.3 Execução da Extração

Conforme descrito no Capítulo 4.

B.4.4 Resolução de Divergências entre os Pesquisadores

Não houve divergências.

B.5 SUMARIZAÇÃO DOS RESULTADOS

B.5.1 Cálculos Estatísticos sobre os Resultados

Não se aplica.

B.5.2 Apresentação dos Resultados em Tabelas:

Vide Tabela 6 do Capítulo 4.

B.5.3 Análise de Sensibilidade

Não se aplica.

B.5.4 Plotagem

Não se aplica.

B5.5 Comentários Finais

- Número de estudos: foram retornados 132 artigos dos quais 10 foram selecionados. Além disso, outros 3 artigos foram incluídos para extração de informações.
- Vieses identificados: o número de fontes de busca utilizadas (quatro); a qualidade dos motores de busca das fontes selecionadas; e a influência do autor na seleção dos artigos e na extração das informações.

- Variação entre revisores: não se aplica.
- Aplicação dos resultados: os resultados servirão de base para conclusão da pesuisa do autor.
- Recomendações: nenhuma.

APÊNDICE C – INTERFACES DESENVOLVIDAS PARA O ESTUDO EXPERIMENTAL SOBRE O MODELO SPIM

Este apêndice contém a descrição detalhada das interfaces desenvolvidas para a realização do estudo experimental sobre o modelo integrado SPIM.

Inicialmente, os pesquisadores pensaram em desenvolver um site que pudesse auxiliar na promoção da palestra sobre o modelo SPIM. Este site foi desenvolvido em PHP, acessando uma base de dados MySQL. O site encontra-se hospedado em um servidor do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Câmpus Bento Gonçalves.

Desta forma, foi enviado um e-mail para as duas turmas de pós-graduação em gestão de projetos informando sobre a palestra, sobre o estudo experimental e sobre o link de acesso: http://napne.bento.ifrs.edu.br/spim/views/index.php. A Figura 85 apresenta a página inicial deste portal.

Figura 85. Portal sobre o estudo experimental do SPIM

Outro fator importante para o desenvolvimento deste portal foi de oferecer um mecanismo que permitisse maior segurança e confiabilidade dos dados informados durante o estudo experimental.

Este portal está divido nas seguintes seções: a) Página inicial, b) Apresentação do Projeto, c) Inscrição no evento, d) Publicações do grupo de pesquisa, e) Contato, f) Formulários do Estudo Experimental. Cada uma destas seções é apresentada a seguir.

C.1 PÁGINA INICIAL - CONVITE

A Figura 86 apresenta a página inicial deste portal contendo o convite para o estudo experimental sobre o modelo SPIM.

Convite

Prezado(a)s,

Gostaríamos de convidá-lo para conhecer o modelo de planejamento integrado **SPIM (Software Planning Integrated Model)** desenvolvido por pesquisadores do Programa de Pós-Graduação em Ciência da Computação da PUCRS. O **SPIM** compreende um modelo para suporte ao gerente de projetos no planejamento e tratamento das atividades relativas aos demais fluxos de atividades da organização (aqui denominados **fluxos organizacionais**).

Durante a execução das atividades de um projeto de software, o gerente de projetos pode não possuir todas as informações relevantes para o projeto. Por exemplo, o gerente de projetos pode precisar contatar o setor de recursos humanos sobre a necessidade de contratação de pessoal para um determinado projeto de desenvolvimento de software. A preparação técnica e a liberação de uma sala ou equipamento testes são outros exemplos cujas atividades não são exclusivas de um projeto em especial, mas de um **fluxo comum da empresa**, compartilhado pelos projetos em andamento e que utiliza recursos não alocados diretamente ao projeto de software. Dessa forma, pode haver também uma relação de dependência entre as atividades pertencentes a estes dois tipos de fluxos de trabalho. A dificuldade para identificar esta interdependência dos fluxos de trabalho durante o planejamento de atividades pode afetar negativamente o projeto, resultando, por exemplo, no **aumento dos custos** e em **atrasos nos prazos** do projeto. O modelo **SPIM** foi concebido considerando a necessidade do gerente de projetos obter acesso às informações pertencentes aos outros departamentos da organização durante o planejamento de projetos de software.

O evento compreende uma apresentação do modelo **SPIM** e a realização de uma **atividade prática (experimento acadêmico)** onde os participantes terão a oportunidade de realizar um exercício baseado em uma situação típica para o problema de gestão de projetos envolvendo os fluxos organizacionais, unindo teoria e prática.

Tal exercício se caracteriza como um estudo experimental, cujos resultados permitirão aos pesquisadores a realização de uma análise sobre as dificuldades impostas aos gerentes de projetos na resolução do problema com e sem o uso do modelo **SPIM**. Os resultados serão encaminhados aos participantes do evento, permitindo-lhes uma

avaliação sobre os resultados do experimento a partir da experiência realizada pelo grupo.

O evento é gratuito e as vagas são limitadas. Estão convidados alunos e professores de graduação e pós-graduação que tenham interesse na área de gerenciamento de projetos. As informações fornecidas serão tratadas de forma absolutamente confidencial. Ao participar, além dos resultados do experimento você terá acesso aos materiais publicados por este grupo de pesquisadores. Os resultados devem ser disponibilizados a partir de novembro de 2011, sendo que estaremos lhe enviando uma comunicação por e-mail.
Para participar, basta clicar aqui e criar uma conta de acesso. A data limite para a cadastro e participação é **26 de outubro de 2011**. Lembramos que as vagas são limitadas. **Cadastre-se já!**

Colabore com o desenvolvimento das pesquisas na área de gerenciamento de projetos no Brasil. **Participe!**

Em caso de dúvidas, entre em contato através do e-mail: mauricio.rosito@acad.pucrs.br

Figura 86. Tela sobre o convite para a palestra do SPIM

C.2 APRESENTAÇÃO DO PROJETO

A Figura 87 apresenta a página que contém a apresentação dos integrantes do grupo de pesquisa sobre a reconfiguração dinâmica de projetos de software.

Apresentação do Projeto
TÍTULO:

- Reconfiguração Dinâmica de Projetos de Desenvolvimento de Software

ÁREAS DE PESQUISA:

- Reconfiguração dinâmica de projetos de software
- Gerenciamento de Projetos de Software
- Inteligência Artificial Aplicada

GRUPOS DE PESQUISA RELACIONADOS:

- SySModE - Grupo de Modelagem e Especificação de Sistemas e Software
- ISEG - Intelligent Systems Engineering Group (http://semanticore.pucrs.br)

PARTICIPANTES DO PROJETO:
Coodenador

- Ricardo Melo Bastos, Doutor

Professor do Programa de Pós-Graduação em Ciência da Computação da PUCRS
E-mail: bastos@pucrs.br - http://lattes.cnpq.br/5205533032581356
Equipe:

- Marcelo Blois Ribeiro, Doutor

Professor do Programa de Pós-Graduação em Ciência da Computação da PUCRS
E-mail: marcelo.blois@pucrs.br - http://lattes.cnpq.br/9863163058520891

- Maurício Covolan Rosito, Mestre

Doutorando do Programa de Pós-Graduação em Ciência da Computação da PUCRS
E-mail: mauricio.rosito@acad.pucrs.br - http://lattes.cnpq.br/3340025175172044

- Daniel Antonio Callegari, Doutor

Professor da Faculdade de Informática da PUCRS
E-mail: daniel.callegari@pucrs.br - http://lattes.cnpq.br/5667898309290802

Figura 87. Tela sobre de apresentação da equipe de pesquisadores

C.3 APRESENTAÇÃO DA PALESTRA

A Figura 88 apresenta a página que contém informações sobre como realizar a inscrição na palestra (e posterior estudo experimental) sobre o modelo integrado SPIM.

Inscrição para a Palestra sobre o mopelo SPIM
Palestrante: Maurício Covolan Rosito (doutorando em Ciência da Computação da PUCRS)
Data: 26/10/2011 das 08:50hrs até às 12:00hrs
Local: Laboratório 411 do Prédio da Faculdade de Informática da PUCRS - Porto Alegre/RS

Público Alvo: Alunos de pós-graduação da PUCRS
Lotação: Até 30 participantes.

Motivação: Projetos de software são muito dinâmicos e demandam recorrentes ajustes dos seus planos de projeto. Esses ajustes podem ser vistos como reconfigurações no cronograma de tarefas, na atribuição de recursos e de outros elementos do projeto. Ainda, durante o planejamento e execução de um projeto de software, deve-se considerar a integração das atividades específicas dos projetos com as atividades que fazem parte de um fluxo de atividades comum à organização. Neste sentido, está sendo proposto um modelo computacional para o suporte a reconfiguração dinâmica de projetos de software em ambientes multiprojetos, considerando o planejamento e o replanejamento de suas atividades, e com ênfase na integração da gerência de projetos com os fluxos organizacionais das empresas. Para avaliação do modelo e materialização da proposta, foi desenvolvido um protótipo de uma ferramenta de software.

Inscrição: Faça sua inscrição aqui.

Figura 88. Tela sobre a inscrição para a palestra

Em seguida, o usuário era convidado a preencher o formulário de inscrição para a palestra (uma vez que se tratava de um evento com lugares limitados), conforme visto na Figura 89.

Figura 89. Formulário de inscrição para a pilastra

C.4 APRESENTAÇÃO DO PROJETO

A Figura 90 apresenta a página que contém informações sobre as publicações deste grupo de pesquisa sobre assuntos relacionados à área de gestão de projetos.

Publicações
Nesta seção é apresentada a produção científica recente deste grupo de pesquisadores relativo a temática desta palestra.

- CALLEGARI, D. Reconfiguração Dinâmica de Projetos de Desenvolvimento de Software. Tese de Doutorado, PUC-RS, Porto Alegre, 2010.
- SCHLÖSSER, R. Gerenciamento Distribuído de Agendas de Recursos para Projetos de Desenvolvimento de Software baseado em Sistemas Multiagentes. Dissertação de Mestrado, PUC-RS Porto Alegre, 2010.
- CALLEGARI, Daniel Antonio ; BASTOS, R. M. . A Multi-Criteria Resource Selection Method for Software Projects using Fuzzy Logic. In: ICEIS 2009 - 11th International Conference on Enterprise Information Systems, 2009, Milan. Enterprise Information Systems. Berlin : Springer-Verlag, 2009. v. LNBIP. p. 376-388.
- CALLEGARI, Daniel Antonio ; Foliatti, F.L. ; BASTOS, R. M. . MRES - Ferramenta para Seleção de Recursos para Tarefas de Projetos de Software via Abordagem Difusa Multicritérios. In: XVI Sessão de Ferramentas - SBES, 2009, Fortaleza. Sessão de Ferramentas 2009 - XVI Sessão de Ferramentas - SBES, 2009. p. 61-66.
- CALLEGARI, D. ; BASTOS, R. M. A Systematic Review of Dynamic Reconfiguration of Software Projects. XXII Simpósio Brasileiro de Engenharia de Software – SBES 2008. Campinas, 2008.

- CALLEGARI; ROSITO; BASTOS; RIBEIRO. An Integrated Model for Managerial and Productive Activities in Software Development. ICEIS 2008 – Int. Conf. on Enterprise Information Systems. 2008.
- M. ROSITO; D. CALLEGARI; R. BASTOS. Gerência de Projetos e Processos de Desenvolvimento de Software: uma proposta de integração. iSys - Revista Brasileira de Sistemas de Informação - PPGI / UNIRIO, v. 1, p. 88-115, 2008.
- ROSITO, M.; CALLEGARI, D.; BASTOS, R. Gerência de Projetos e Processos de Desenvolvimento de Software: uma proposta de integração. SBSI – IV Simpósio Brasileiro de Sistemas de Informação 2008.
- ROSITO, M. Um modelo de Integração entre a Gerência de Projetos e o Processo de Desenvolvimento de Software. Dissertação de Mestrado, PUC-RS, Porto Alegre, 2008.
- CALLEGARI, D.; BASTOS, R. Project Management and Software Development Processes: Integrating RUP and PMBOK. ICSEM – International Conference on Systems Engineering and Modeling, 2007.
- ROSITO, Maurício Covolan; CALLEGARI, Daniel. A.; BASTOS, Ricardo Melo. Metamodelos de processos de desenvolvimento de software: Um estudo comparativo. In: III Simpósio Brasileiro de Sistemas de Informação, 2006.
- BASTOS, R. M. ; OLIVEIRA, Flávio Moreira de ; OLIVEIRA, José Palazzo Moreira de . Autonomic computing approach for resource allocation. Expert Systems with Applications, Estados Unidos, v. 28, n. 1, p. 9-19, 2005.

Figura 90. Tela sobre as publicações do grupo de pesquisa

C.5 APRESENTAÇÃO DO PROJETO

A Figura 91 apresenta a página que para contato com os pesquisadores deste trabalho.

Contato

Em caso de dúvidas, entre em contato através do e-mail: mauricio.rosito@acad.pucrs.br

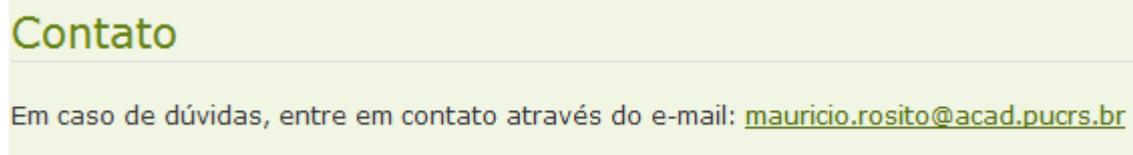



Figura 91. Tela para contato com os pesquisadores

C.6 ÁREA RESTRITA DO PORTAL

Uma vez que os alunos receberam sua confirmação de inscrição no estudo experimental sobre o modelo SPIM, eles puderam ter acesso à área restrita do site (ver Figura 92).

Figura 92. Tela de acesso restrito do portal

Neste espaço, os usuários poderiam atualizar seus dados cadastrais, obter informações sobre outras palestras relacionadas a esta área de estudo e, principalmente, ter acesso aos formulários relacionados ao estudo experimental (Figura 93). Maiores detalhes sobre os formulários do estudo experimental são apresentados no ApêndiceE.

Estudo experimental sobre o SPIM
Participante: Teste Usuário
Orientações: Por favor, preencha o questionário de avaliação do perfil dos entrevistados. Em seguida, preencha os cinco cenários de avaliação do modelo intrerado SPIM que irão simular situações em projetos de desenvolvimento de software. Ao final, preencha o questionário de avaliação do modelo SPIM.
Sua colaboração é muito importante para o sucesso dessa pesquisa acadêmica.
Por favor, complete todos os passos abaixo para efetivar sua participação neste experimento.
Passos:

- Questionário de avaliação do perfil dos entrevistados
- Cenários de avaliação do modelo integrado SPIM
    1. Cenário 1
    2. Cenário 2
    3. Cenário 3
    4. Cenário 4
    5. Cenário 5
- Questionário de avaliação do modelo SPIM

Observação: Os resultados desta pesquisa serão disponibilizados para download de materiais em breve.
Obrigado por sua participação.

Figura 93. Tela de apresentação do estudo experimental

APÊNDICE D – Metodologia de Desenvolvimento do Protocolo de Avaliação do SPIM

Este apêndice contém a descrição detalhada da metodologia de desenvolvimento do protocolo de avaliação do SPIM.

D.1 OBJETIVO

Avaliar os conceitos advindos do modelo integrado SPIM no que diz respeito à sua aceitação e aplicabilidade, de acordo com o ponto de vista dos estudantes de pós-graduação em gestão de projetos de software.

D.2 CARACTERÍSTICA-CHAVE DO MÉTODO DE PESQUISA

Este é o roteiro para o desenvolvimento e aplicação de um instrumento de pesquisa baseado em um estudo experimental.

D.3 ORGANIZAÇÃO DESSE PROTOCOLO DE AVALIAÇÃO

D.3.1 Procedimentos

Tabela 22: Procedimentos para o desenvolvimento do protocolo de avaliação

A. Reuniões para levantamento das questões e estruturação do roteiro do estudo	
Participantes:	Maurício Covolan Rosito Prof. Dr. Ricardo Melo Bastos
Local:	PPGCC (Programa de Pós-Graduação em Ciência da Computação – FACIN – PUCRS)
Datas:	Abril de 2011
B. Validação de face e conteúdo	
Participante:	Prof. Dr. Marcelo Blois Ribeiro
Local:	PPGCC (Programa de Pós-Graduação em Ciência da Computação – FACIN – PUCRS)
Data:	Maio de 2011
C. Adequação do roteiro do experimento com base na validação de face e conteúdo	
Participante:	Maurício Covolan Rosito
Local:	PPGCC (Programa de Pós-Graduação em Ciência da Computação – FACIN – PUCRS)
Data:	Maio de 20011
D. Pré-Teste	
Participantes:	Maurício Covolan Rosito Msc. Rogério Tessari
Local:	Instituto Federal do Rio Grande do Sul (IFRS)
Data:	Junho de 2011

E. Adequação do roteiro de entrevista com base no pré-teste	
Participante:	Maurício Covolan Rosito
Local:	PPGCC
Data:	Junho de 2011
F. Aplicação do Instrumento	
Tipo:	Experimento
Entrevistador:	Maurício Covolan Rosito
Datas:	04/11/2011 e 26/11/2011
Participantes:	36

D.3.2 Outros Recursos Utilizados

- **Sistema Computacional e Estatístico:**
  - Statistical Package for Social Sciences (SPSS).
- **Recursos Materiais:**
  - Um auditório da PUCRS para a palestra inicial sobre o SPIM;
  - Um auditório do IBGEN para a palestra inicial sobre o SPIM;
  - Laboratório com 30 máquinas instaladas com software SPIT na PUCRS.
  - Laboratório com 30 máquinas instaladas com software SPIT na IBGEN.

D.4 PLANEJAMENTO DO EXPERIMENTO

D.4.1 Levantamento de Objetivos e Hipóteses do Experimento

Objetivo Global

Comparar, no Processo Unificado, a precisão e o esforço do modelo de planejamento integrado SPIM (*Software Planning Integrated Model*) em relação ao modelo tradicional de planejamento de projetos de software.

Objetivos do Estudo Experimental

1. Unidade experimental (*experimental unit*):
   - Projetos de Desenvolvimento de Software.
2. Fatores ou variações provocadas (*factors*):
   - Métodos de planejamento de projetos de desenvolvimento de software.
3. Tratamentos:
   - Método tradicional de planejamento de atividades;
   - Método de planejamento integrado de atividades com o SPIM.
4. Parâmetros (parameters):
   - Complexidade da tarefa ilustrada no cenário;
   - Ordem de execução dos cenários.

5. Variáveis de Resposta (response variables - RVs):
   - Esforço;
   - Precisão.
6. Sujeitos Experimentais (experimental subjects):.
   - Alunos de pós-graduação com experiência anterior em projetos de desenvolvimento de software.
7. Objeto: Cinco cenários simulando situações em projetos de software.
8. Variáveis de bloqueio (blocking variables):
   - Experiência em gestão de projetos.

Objetivo da Medição

Para que as variáveis de resposta sejam medidas é necessário que tenhamos métricas básicas que relacionadas, possam caracterizar o valor das variáveis de resposta. Neste item, optamos pelo o uso da técnica GQM para levantar estas métricas.

Questões

1. O esforço para realizar o planejamento das atividades de projetos de software utilizando o modelo integrado SPIM é igual ao esforço para realizar o planejamento das atividades segundo o modelo tradicional?

2. A precisão no planejamento do cronograma de projetos de software com relação à atribuição de prazos e recursos, pensando na integração com os fluxos organizacionais, através do modelo integrado SPIM é igual à precisão para realizar o planejamento segundo o modelo tradicional?

Métricas

A métrica associada à Questão 1 corresponde ao esforço medido pela relação do tempo gasto em minutos por cada participante durante a realização do planejamento das atividades do projeto de software em cada abordagem.

A métrica relacionada à Questão 2 corresponde à precisão com relação à atribuição de prazos e recursos nas atividades do cronograma do projeto utilizando cada uma das abordagens evitando, desta forma, a ocorrência de determinados tipos de riscos no projeto. Por precisão, foi definido em nosso estudo como sendo a razão entre a pontuação feita pelos participantes e a pontuação total possível, de acordo com um gabarito.

Caracterização Formal das Hipóteses do Estudo

Nesta seção o objetivo é associar as hipóteses informais feitas anteriormente com as métricas levantadas, formulando as hipóteses que guiarão a execução do experimento. As hipóteses informais (em língua natural) devem ser traduzidas em indicadores objetivos (numéricos) para verificação estatística da sua validade. Devido a natureza dos testes estatísticos, formulamos nossas hipóteses em função de uma hipótese nula que é aquela que indica que a variância ocorrida em um fenômeno foi aleatória e, portanto, a introdução de um tratamento novo em um fator não ocasionou influência significativa.

Das hipóteses foram definidas informalmente:

1. É necessário um esforço para o planejamento de projetos de software. Sugere-se que o esforço para realizar o planejamento das atividades do projeto de software utilizando o modelo integrado SPIM é igual ao esforço para realizar o planejamento das atividades segundo o modelo tradicional.

2. Sugere-se que a precisão no planejamento do cronograma de projetos de software com relação à atribuição de prazos e recursos, pensando na integração com os fluxos organizacionais, através do modelo integrado SPIM é igual à precisão para realizar o planejamento segundo o modelo tradicional.

D.4.2 Seleção Das Variáveis

Variáveis Independentes

Assumiram-se como variáveis independentes:

- Experiência do time de gestão (variável de bloqueio);
- Métodos para o planejamento das atividades de projetos de software;

Variáveis Dependentes

Assumiram-se como variáveis dependentes:

- Esforço para o planejamento das atividades pertencentes aos projetos de software;
- Precisão através da razão entre a pontuação feita pelos participantes utilizando determinada técnica e a pontuação total possível, de acordo com um gabarito.

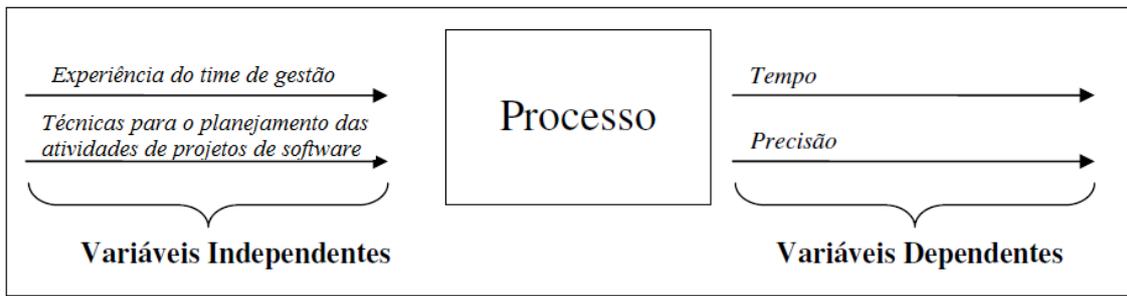



Figura 94. Variáveis independentes e dependentes do estudo experimental

A Tabela 23 sumariza as escalas para cada variável considerada.

Tabela 23: Escalas das variáveis dependentes e independentes

Variáveis	Nome	Escala
Dependentes	Tempo	Razão
	Precisão	Razão
Independentes	Métodos para o planejamento das atividades de projetos de software	Nominal

D.5 PLANEJAMENTO DO EXPERIMENTO

D.5.1 Caracterização Detalhada do Contexto

Para a condução do experimento de sobre o planejamento de projetos de software, foi escolhido o contexto de duas instituições de ensino superior (IBGEN e PUCRS), abordagem que permite diminuir riscos e custos não previstos no escopo da pesquisa neste momento. Dentro das dimensões apresentadas, caracterizam-se:

- Processo: será utilizada à abordagem In-vitro, na qual o conjunto de participantes executará o experimento em um ambiente controlado. Este experimento não se dará durante o planejamento de um projeto de software na indústria.
- Participantes: o experimento será conduzido por alunos de pós-graduação da Faculdade de Informática da PUCRS e do IBGEN;
- Realidade: o problema estudado corresponde a diferentes cenários simulando situações em projetos de desenvolvimento de software;
- Generalidade: o experimento é específico e com validade apenas no escopo do presente estudo.

D.5.2 Seleção dos Indivíduos

A população definida para o experimento é formada por alunos do curso de pós-graduação em gestão de projetos da Faculdade de Informática da PUCRS e do IBGEN. Será um total de trinta e seis alunos

Será utilizada uma amostragem não probabilística para seleção dos indivíduos:

- Amostragem por conveniência: serão escolhidas as pessoas mais convenientes para o experimento;

D.5.3 Princípios Observados para o Projeto do Experimento

Dentre os princípios genéricos para o projeto do experimento, caracterizamos:

- Aleatoriedade: a aleatoriedade será utilizada para definir quais participantes irão executar cada abordagem de planejamento de projetos de software (tradicional ou com auxílio do modelo SPIM);
- Balanceamento: este princípio será utilizado em nosso experimento para que cada proposta de planejamento de projetos de software seja executada pela mesma quantidade de participantes.

D.5.4 Tipo de Experimento e Definição das Unidades Experimentais

Para cada hipótese, serão utilizadas as seguintes notações:

- μ_{trad} Planejamento de atividades de projetos de software tradicional;
- μ_{SPIM} Planejamento de atividades de projetos de software com apoio do modelo SPIM.

O tipo de projeto apresentado procura investigar se μ_{trad}possui o mesmo esforço e precisão que μ_{SPIM}. Para este fim, será utilizada uma abordagem chamada um fator com dois tratamentos. O fator, neste experimento, consiste na técnica que será utilizada e os tratamentos consistem no planejamento tradicional e o planejamento integrado de projetos de software. Por motivos de projeto e complexidade, utilizaremos o tipo de projeto completamente aleatório, onde cada participante executará apenas uma abordagem definida aleatoriamente.

D.5.5 Instrumentação

Nesta pesquisa, a aquisição dos dados será realizada por meio de um questionário (Apêndice E) elaborado em conformidade com Rea & Parker (2005). Assim, a elaboração do questionário foi realizada de acordo com os seguintes passos:

- coleta de dados preliminares a respeito do tema e da população alvo da pesquisa;

- discussão em grupo sobre as questões;
- elaboração do rascunho do questionário;
- realização de um pré-teste;
- revisão do instrumento baseado nas observações obtidas no pré-teste; e
- delineamento do questionário final.

Na primeira etapa será realizado o levantamento bibliográfico e o estudo do referencial teórico que permitirá aprofundar os conhecimentos sobre riscos e eventos durante o planejamento em projetos de software e sobre processos de gerência de projetos. Além disso, será definida a população alvo desta pesquisa (estudantes de pós-graduação, neste caso). Em seguida, foram realizadas reuniões entre o pesquisador e o professor orientador para o levantamento das questões e a estruturação do roteiro de entrevistas. Estas reuniões resultarão na elaboração de um rascunho do questionário necessário para a aquisição dos dados desta pesquisa. Posteriormente, será realizada a validação de face e conteúdo do protocolo de avaliação da pesquisa por um pesquisador sênior. Com base nesta validação o protocolo poderá sofrer pequenas correções. A seguir será realizado o pré-teste com um pesquisador sênior, onde foi possível realizar os últimos ajustes no roteiro de entrevistas.

D.5.6 Considerações sobre a Validade do Experimento

Validade interna

Serão avaliados alguns critérios, tais como:

- Histórico: a data de aplicação do experimento será criteriosamente definida, evitando períodos nos quais os participantes possam sofrer influências externas;
- Maturação: durante o treinamento, serão utilizadas técnicas de motivação para incentivar positivamente os participantes;
- Seleção dos grupos: será utilizada uma abordagem para nivelar o conhecimento dos participantes através de um treinamento sobre as técnicas. A execução das atividades será individual;
- Difusão: durante o treinamento, será desenvolvida uma motivação que não incentive interação entre os participantes. Adicionalmente, haverá um policiamento durante a experimentação para evitar este tipo de interação;

Validade externa

Para esta avaliação, será adotada a interação da seleção, ou seja, os participantes que foram selecionados possuem um perfil apto aos tratamentos do experimento, apresentando, em sua maioria, conhecimento prévio sobre processo de desenvolvimento de software e gestão de projetos, além de experiência em indústria de software.

Validade de construção

Durante nosso experimento, serão avaliados:

- Inadequada explicação pré-operacional: consiste na explicação operacional do experimento, visando maturar a forma na qual será definida a extração dos dados;
- Adivinhação de hipóteses: devido ao fato dos participantes serem humanos, é possível sua interação com o experimento, sugerindo novas hipóteses e exercitando a criatividade. É importante manter o foco no estudo planejado;
- Expectativas do condutor do experimento: ao se conduzir um experimento, o responsável pode exercer influências sobre as variáveis envolvidas e sobre o material elaborado. Durante a presente proposta, todo o material utilizado será previamente avaliado por outro responsável.

Validade da conclusão

Serão avaliadas as seguintes perspectivas:

- Manipulação dos dados: como os dados resultantes do experimento serão manipulados pelo pesquisador, é possível que os mesmos sofram algumas variações, tal como o coeficiente de significância para validação dos resultados;
- Confiabilidade das medidas: esta perspectiva sugere que medidas subjetivas possam ser influenciadas pelo pesquisador. Em nossa proposta, as medidas foram objetivamente definidas, não dependendo do critério humano;
- Confiabilidade na implementação dos tratamentos: consiste no risco em que diferentes participantes possam implementar de forma distinta os processos estabelecidos pelo experimento. Este risco não será evitado em nosso estudo, visto que não se pode interferir no caráter subjetivo do planejamento de atividades de projetos de software. Possivelmente, diferentes participantes definirão planejamentos distintos;

- Configurações do ambiente do experimento: consiste nas interferências externas do ambiente que podem influenciar os resultados durante a execução do experimento. O experimento será executado laboratórios isolados, onde será proibida a interação externa como celulares, saídas, etc.;
- Heterogeneidade aleatória dos participantes: a escolha de diferentes participantes com diferentes experiências pode exercer um risco na variação dos resultados.

D.5.7 Aspectos para a Execução do Experimento

Para preparar a execução do experimento, atentou-se para:

- Consenso com o experimento: Durante a experimentação, a preparação dos participantes deverá fornecer o embasamento necessário sobre o experimento;
- Resultados sensitivos: é possível que o resultado obtido pelo experimento se influencie por questões pessoais, como a sensibilidade dos participantes por estarem sendo avaliados. Será adotada uma postura de anonimato dos participantes em toda a descrição da experimentação.

Com relação à instrumentação, todas as variáveis e os recursos devem ser criteriosamente estabelecidos antes da execução do experimento. Deve ser feito um treinamento específico para cada grupo (planejamento tradicional e planejamento integrado SPIM) em cada local de aplicação (PUCRS e IBGEN), contextualizando os objetivos, a técnica, a motivação e o procedimento técnico para condução do experimento. Para coleta de dados será usado um conjunto de formulários foram definidos para este fim (ApêndiceE). Outro critério a ser considerado é a questão do anonimato, onde os nomes dos participantes não serão registrados.

APÊNDICE E – PROTOCOLO DE AVALIAÇÃO DO SPIM

Este apêndice contém a descrição detalhada do protocolo de avaliação do SPIM.

E.1 QUESTIONÁRIO DE AVALIAÇÃO DO PERFIL DOS ENTREVISTADOS

Veja abaixo o questionário de avaliação do perfil dos entrevistados.

Perfil do Entrevistado
1. Nome: 2. E-mail: 3. Idade: 4. Grau de Escolaridade: 5. Instituição de Ensino: 6. Informe seu tempo de experiência em atividades relacionadas a gestão de projetos: ☐ até 1 ano ☐ entre 1 e 2 anos ☐ entre 2 e 5 anos ☐ entre 5 e 10 anos ☐ mais de 10 anos 7. Informe seu tempo de experiência em atividades relacionadas a processos de desenvolvimento de software: ☐ até 1 ano ☐ entre 1 e 2 anos ☐ entre 2 e 5 anos ☐ entre 5 e 10 anos ☐ mais de 10 anos 8. Já realizou algum treinamento referente à Gerência de Projetos? 9. Como você qualificaria seu conhecimento relacionado à Gerência de Projetos? ☐ Nenhum ☐ Pouco ☐ Moderado ☐ Avançado 10. Como você qualificaria seu conhecimento relacionado à processos de desenvolvimento de software? ☐ Nenhum ☐ Pouco ☐ Moderado ☐ Avançado

E.2 CENÁRIO 1

Veja abaixo a descrição das atividades propostas para o cenário 1.

Cenário 1
Participante: Teste Usuário **Download:** Clique aqui para fazer download do cenário 1 **Descrição do Cenário 1:**

O papel do *stakeholder* envolvido deve ser compatível com o tipo de atividade (gerencial ou produtivo)

Recursos envolvidos neste projeto:

- Carlos Viana: Gerente de Sistemas.
- Mário Silva: Gerente de Projetos.
- Ana Valente: Projetista de Sistemas.
- João Souza: Analista de Sistema.
- Mácio Oliveira: Desenvolvedor Senior.
- Daniel Carvalho: Desenvolvedor Junior.
- Diego Moraes: DBA Senior.
- Samanta Pinheiro: Analista de Testes.
- Joana Santos: Testadora.
- Sala de Reunião 1.
- Sala de Reunião 2.

Orientações:

1. Abra o arquivo Cenário1;
2. Caso você esteja usando o modelo SPIM, associe cada recurso a um papel de acordo com a listagem acima. Para realizar este operação, clique no menu "Exibir" do MS Project e selecione a opção "Planilha de Recursos". Ao finalizar, clique no menu "Exibir" do MS Project e selecione a opção "Gantt de Controle" para exibir as atividades do projeto.
3. Sem alterar as datas e dados de precedência entre as atividades e de acordo com a tabela de recursos acima, procure associar a cada atividade um recurso apropriado.

1. Deve-se procurar respeitar que o papel do *stakeholder* envolvido deve ser compatível com o tipo de atividade (gerencial ou produtiva)
2. Caso você esteja usando o modelo SPIM, associe cada atividade do cronograma como sendo produtiva ou gerencial

As atividades consideradas gerenciais para este projeto são as seguintes:

- Reunião de Acompanhamento Gerencial
- Elaborar documento de Avaliação / Revisão
- Organizar e Conduzir a Reunião de Kick-Off
- Elaborar e Revisar Caso de Desenvolvimento
- Revisão de estimativas
- Estimar Custos e Recursos
- Refinar Planejamento do Projeto
- Conduzir Reunião de Avaliação / Revisão
- Encerrar Projeto

Todas as outras atividades são consideradas produtivas.

Verifique se todas as atividades possuem um recurso associado. Caso você esteja usando o modelo SPIM, ative os filtros listados abaixo e faça a validação do projeto:

- Tipo de papel (gerencial ou produtivo)
- Tipo de atividade (gerencial ou produtiva)

Salve o arquivo com o nome: Cenario1_SeuNome

Respostas

Arquivo: : [ ] Salvar

Veja abaixo uma parte do projeto a ser avaliado no cenário 1.

		Nome da tarefa	Duração	Início	Término	Predecessor
1		**Projeto - Cenário 1**	**35,25 dias**	**29/11/2011**	**17/01/2012**	
2		**Iniciação**	**12 dias**	**29/11/2011**	**14/12/2011**	
3		**Monitoramento e Controle**	**1 dia**	**29/11/2011**	**29/11/2011**	
4		Reunião de Acompanhamento - Gerencial	1 dia	29/11/2011	29/11/2011	
5		**Plano de Projeto**	**11 dias**	**30/11/2011**	**14/12/2011**	
6		Elaborar documento de Avaliação / Revisão	2 dias	30/11/2011	01/12/2011	4
7		Organizar e Conduzir a Reunião de Kick-Off	1 dia	02/12/2011	02/12/2011	6
8		Elaborar e Revisar Caso de Desenvolvimento	3 dias	05/12/2011	07/12/2011	7
9		Revisão de estimativas	3 dias	08/12/2011	12/12/2011	8
10		Estimar Custos e Recursos	2 dias	13/12/2011	14/12/2011	9
11		**Elaboração**	**7 dias**	**15/12/2011**	**23/12/2011**	
12		**Revisão do Plano de Projeto**	**5 dias**	**15/12/2011**	**21/12/2011**	
13		Refinar Planejamento do Projeto	5 dias	15/12/2011	21/12/2011	10
14		**Área de Engenharia**	**6 dias**	**15/12/2011**	**22/12/2011**	
15		Definir Arquitetura	3 dias	15/12/2011	19/12/2011	10
16		Análise dos documentos de ERS e Visão - Ambientação no projeto	3 dias	15/12/2011	19/12/2011	10
17		Colocar Artefatos sob Rastreabilidade	2 dias	20/12/2011	21/12/2011	16
18		Teste da Arquitetura	3 dias	20/12/2011	22/12/2011	15
19		**Revisar Projeto e Aprovações**	**1 dia**	**23/12/2011**	**23/12/2011**	
20		Conduzir Reunião de Avaliação / Revisão	1 dia	23/12/2011	23/12/2011	18;13;17
21		**Construção**	**10,25 dias**	**26/12/2011**	**09/01/2012**	
22		**Arquitetura**	**3 dias**	**26/12/2011**	**28/12/2011**	
23		Definição do processo de implementação	1 dia	26/12/2011	26/12/2011	20
24		Modelar Banco de Dados	2 dias	27/12/2011	28/12/2011	23
25		**Especificação Requisitos - Release 1.0**	**7 dias**	**29/12/2011**	**06/01/2012**	
26		RNF6 - Front End em Inglês - DSV	2 dias	29/12/2011	30/12/2011	24
27		RNF8 - Construção do Front End - ETR	4 dias	29/12/2011	03/01/2012	24
28		RF227 - Manter Perfis - ERR	3 dias	04/01/2012	06/01/2012	27
29		**Especificar Teste**	**2 dias**	**27/12/2011**	**28/12/2011**	
30		Elaborar Especificação de Plano de Teste (Iniciação e Elaboração)	2 dias	27/12/2011	28/12/2011	23

Figura 95. Projeto para o cenário 1

E.2 CENÁRIO 2

Veja abaixo a descrição das atividades propostas para o cenário 2.

Cenário 2
Participante: Teste Usuário
Download: Clique aqui para fazer download do cenário 2
Glossário:
Fluxos Organizacionais:

- RH_Contratar: Fluxo Organizacional responsável pela contratação de pessoas.
- RH_Treinamento: Fluxo Organizacional responsável pelas atividades de treinamento da equipe.
- Financeiro_Comprar: Fluxo Organizacional responsável pela aquisição de equipamentos.
- Negocios_Parceria: Fluxo Organizacional responsável pela realização de parcerias e contratação de empresas terceirizadas.

Descrição do Cenário 2:
Necessidade de aquisição de um novo hardware ou software durante o projeto.
Orientações:

1. Abra o arquivo Cenário2;
2. Durante a execução da atividade "Revisão de estimativas", no dia 17/11/2011, você se depara com a necessidade de comprar um novo servidor para hospedar o código fonte

do projeto. Esta é uma atividade pertencente ao fluxo organizacional do setor financeiro, cujo tempo de duração é de 4 (quatro) dias.

3. Com base nesta situação, sem alterar a ordem das atividades no projeto, defina qual é o prazo máximo para fazer esta solicitação ao setor administrativo **(Digite sua resposta)**.

4. Caso você esteja usando o modelo SPIM:

- Atribua uma relação de dependência entre as atividades do projeto de software e as atividades pertencentes ao fluxo organizacional: Clique na atividade que depende do resultado da realização do fluxo organizacional para que seja iniciada. Após, selecione a aba "Campos Personalizados". No campo personalizado chamado "Fluxos Organizacionais", informe o nome do fluxo organizacional (tenha atenção para que tenha exatamente o mesmo nome definido na tabela acima).

- Ative o filtro listado abaixo e faça a validação do projeto: Associação com os Fluxos Organizacionais.

Salve o arquivo com o nome: Cenário2_SeuNome

Respostas

Resposta: ■ 10

Arquivo: : ■ Salvar

Veja abaixo uma parte do projeto a ser avaliado no cenário 2.

		Nome da tarefa	Duração	Início	Término	Predecessoras	Nomes dos recursos
1		**Projeto - Cenário 2**	**23 dias?**	**15/11/2011**	**15/12/2011**		
2		**Iniciação**	**7 dias**	**15/11/2011**	**23/11/2011**		
3		**Revisão do Plano de Projeto**	**7 dias**	**15/11/2011**	**23/11/2011**		
4		Elaborar documento de Avaliação	1 dia	15/11/2011	15/11/2011		Fulano1
5		Revisão de estimativas	3 dias	16/11/2011	18/11/2011	4	Fulano1
6		Estimar custos e recursos	2 dias	21/11/2011	22/11/2011	5	Fulano2
7		Finalizar Plano de Projeto	1 dia	23/11/2011	23/11/2011	6	
8		**Elaboração**	**8 dias**	**24/11/2011**	**05/12/2011**		
9		**Revisão do Plano de Projeto**	**1 dia**	**24/11/2011**	**24/11/2011**		
10		Refinar Planejamento do Projeto	1 dia	24/11/2011	24/11/2011	7	Fulano2
11		**Área de Engenharia**	**7 dias**	**25/11/2011**	**05/12/2011**		
12		Definir Arquitetura	4 dias	29/11/2011	02/12/2011	10	Fulano3
13		Análise dos documentos de ERS e Visão	2 dias	25/11/2011	28/11/2011	10	Fulano4
14		Colocar Artefatos sob Rastreabilidade	1 dia	29/11/2011	29/11/2011	13	Fulano3
15		Teste da Arquitetura	1 dia	30/11/2011	30/11/2011	14	Fulano5
16		Finalizar Versão	1 dia	05/12/2011	05/12/2011	12;15	
17		**Construção**	**6 dias**	**06/12/2011**	**13/12/2011**	**8**	
18		**Arquitetura**	**3 dias**	**06/12/2011**	**08/12/2011**		
19		Definição do processo de implantação	1 dia	06/12/2011	06/12/2011	8	Fulano6
20		Modelar Banco de Dados	2 dias	07/12/2011	08/12/2011	19	Fulano7
21		**Especificação de Requisitos**	**3 dias**	**09/12/2011**	**13/12/2011**		
22		Desenvolvimento	2 dias	09/12/2011	12/12/2011	20	Fulano8
23		Finalizar fase de construção	1 dia	13/12/2011	13/12/2011	22	
24		**Transição**	**2 dias?**	**14/12/2011**	**15/12/2011**		
25		**Fechar Projeto**	**2 dias?**	**14/12/2011**	**15/12/2011**		
26		Reunião para Encerrar Projeto	1 dia	14/12/2011	14/12/2011	23	Fulano2

Figura 96. Projeto para o cenário 2

E.3 CENÁRIO 3

Veja abaixo a descrição das atividades propostas para o cenário 3.

Cenário 3

Participante: Teste Usuário
Download: Clique aqui para fazer download do cenário 3
Fluxos Organizacionais:

- RH_Contratar: Fluxo Organizacional responsável pela contratação de pessoas.
- RH_Treinamento: Fluxo Organizacional responsável pelas atividades de treinamento da equipe.
- Financeiro_Comprar: Fluxo Organizacional responsável pela aquisição de equipamentos.
- Negocios_Parceria: Fluxo Organizacional responsável pela realização de parcerias e contratação de empresas terceirizadas.

Descrição do Cenário 3:
O pessoal mais qualificado está doente e não disponível nos momentos críticos.
Orientações:

1. Abra o arquivo Cenário3;
2. Durante a execução da atividade "Revisão de estimativas", no dia 21/09/2011, você se depara com um atestado médico informando que o único BDA Sênior da equipe está doente e deve ficar fora do projeto por 2 meses. Desta forma, você deve entrar em contato com o setor recursos humanos informando a necessidade de contratar um novo DBA Sênior. A contratação de pessoas é uma atividade pertencente ao fluxo organizacional do setor de recursos humanos, cujo tempo de duração é de 5 (cinco) dias.
3. Com base nesta situação, sem alterar a ordem das atividades no projeto, defina qual é o prazo máximo limite para fazer esta solicitação ao setor de recursos humanos **(Digite sua resposta)**.
4. Caso você esteja usando o modelo SPIM:

- Atribua uma relação de dependência entre as atividades do projeto de software e as atividades pertencentes ao fluxo organizacional: Clique na atividade que depende do resultado da realização do fluxo organizacional para que seja iniciada. Após, selecione a aba "Campos Personalizados". No campo personalizado chamado "Fluxos Organizacionais", informe o nome do fluxo organizacional (tenha atenção para que tenha exatamente o mesmo nome definido na tabela acima).
- Ative o filtro listado abaixo e faça a validação do projeto: Associação com os Fluxos Organizacionais.

Salve o arquivo com o nome: Cenário3_SeuNome

Respostas
Resposta: ■ 20
Arquivo: ■ Salvar

Veja abaixo uma parte do projeto a ser avaliado no cenário 3.

		Nome da tarefa	Duração	Início	Término	Predecessoras
7		**Elaboração**	**6 dias?**	**30/09/2011**	**07/10/2011**	
8		**Área de Engenharia**	**5 dias**	**30/09/2011**	**06/10/2011**	
9		Definir Arquitetura	3 dias	30/09/2011	04/10/2011	6
10		Análise dos documentos de ERS e Visão - An	2 dias	30/09/2011	03/10/2011	6
11		Colocar Artefatos sob Rastreabilidade	1 dia	04/10/2011	04/10/2011	10
12		Teste da Arquitetura	2 dias	05/10/2011	06/10/2011	9
13		Finalizar Fase de Elaboração	1 dia?	07/10/2011	07/10/2011	11;12
14		**Construção**	**12 dias?**	**10/10/2011**	**25/10/2011**	
15		**Arquitetura**	**4 dias**	**10/10/2011**	**13/10/2011**	
16		Definição do processo de implementação	1 dia	10/10/2011	10/10/2011	13
17		Modelar Banco de Dados	2 dias	12/10/2011	13/10/2011	16
18		**Especificação Requisitos - Release 1.0**	**7 dias**	**14/10/2011**	**24/10/2011**	
19		RNF6 - Front End em Inglês - DSV	2 dias	14/10/2011	17/10/2011	17
20		RNF8 - Construção do Front End - ETR	4 dias	14/10/2011	19/10/2011	17
21		RF227 - Manter Perfis - ERR	3 dias	20/10/2011	24/10/2011	20
22		**Especificar Teste**	**2 dias**	**11/10/2011**	**12/10/2011**	
23		Elaborar Especificação de Plano de Teste (Inic	2 dias	11/10/2011	12/10/2011	16
24		**Executar Testes**	**2 dias**	**13/10/2011**	**14/10/2011**	
25		Executar Plano de Teste	2 dias	13/10/2011	14/10/2011	23
26		Finalizar Fase de Construção	1 dia?	25/10/2011	25/10/2011	19;21;25
27		**Transição**	**2 dias**	**26/10/2011**	**27/10/2011**	
28		**Integrar Sistema**	**2 dias**	**26/10/2011**	**27/10/2011**	
29		Preparação e Validação do Ambiente	1 dia	26/10/2011	26/10/2011	14
30		Revisar Projeto e Aprovações	1 dia	27/10/2011	27/10/2011	29
31		**Disponibilização da release 1.0 - PRD**	**3 dias**	**27/10/2011**	**31/10/2011**	
32		Acompanhamento do projeto	2 dias	27/10/2011	28/10/2011	29
33		Teste de Aceitação	3 dias	27/10/2011	31/10/2011	29
34		**Fechar Projeto**	**2 dias?**	**01/11/2011**	**02/11/2011**	
35		Reunião para Encerrar Projeto	1 dia	01/11/2011	01/11/2011	32;33
36		Finalizar Projeto	1 dia?	02/11/2011	02/11/2011	30;35

Figura 97. Projeto para o cenário 3

E.4 CENÁRIO 4

Veja abaixo a descrição das atividades propostas para o cenário 4.

Cenário 4
Participante: Teste Usuário **Download:** Clique aqui para fazer download do cenário 4 **Fluxos Organizacionais:** - RH_Contratar: Fluxo Organizacional responsável pela contratação de pessoas. - RH_Treinamento: Fluxo Organizacional responsável pelas atividades de treinamento da equipe. - Financeiro_Comprar: Fluxo Organizacional responsável pela aquisição de equipamentos. - Negocios_Parceria: Fluxo Organizacional responsável pela realização de parcerias e contratação de empresas terceirizadas. **Descrição do Cenário 4:** O pessoal mais qualificado está doente e não disponível nos momentos críticos. *Orientações:* 1. Abra o arquivo Cenário4; 2. Durante a execução da atividade "Start-Up do Projeto", no dia 20/09/2011, você recebe a notícia que a empresa contratou dois novos Desenvolvedores Junior para integrarem sua equipe. Entretanto, eles precisam fazer um treinamento obrigatório para entrar em seu projeto. Desta forma, você deve entrar em contato com o setor de recursos humanos sobre a necessidade de fazer o treinamento para estes 2 novos desenvolvedores. A coordenação de treinamento de pessoas é uma atividade pertencente

ao fluxo organizacional do setor de recursos humanos, cujo tempo de duração é de 3 (três) dias.

3. Com base nesta situação, sem alterar a ordem das atividades no projeto, defina qual é o prazo máximo para fazer esta solicitação ao setor de recursos humanos **(Digite sua resposta)**.

4. Caso você esteja usando o modelo SPIM:

- Atribua uma relação de dependência entre as atividades do projeto de software e as atividades pertencentes ao fluxo organizacional: Clique na atividade que depende do resultado da realização do fluxo organizacional para que seja iniciada. Após, selecione a aba "Campos Personalizados". No campo personalizado chamado "Fluxos Organizacionais", informe o nome do fluxo organizacional (tenha atenção para que tenha exatamente o mesmo nome definido na tabela acima).
- Ative o filtro listado abaixo e faça a validação do projeto: Associação com os Fluxos Organizacionais.

Salve o arquivo com o nome: Cenário4_SeuNome

Respostas

Resposta: ■ 50

Arquivo: ■ Salvar

Veja abaixo uma parte do projeto a ser avaliado no cenário 4.

		Nome da tarefa	Duração	Início	Término	Predecessoras	Nomes dos recursos
15		**Arquitetura**	**3 dias**	**10/10/2011**	**12/10/2011**		
16		Definição do prc	1 dia	10/10/2011	10/10/2011	13	Fulano 2
17		Modelar Banco	2 dias	11/10/2011	12/10/2011	16	Fulano 5
18		**Especificação Re**	**7 dias**	**13/10/2011**	**21/10/2011**		
19		RNF6 - Front En	3 dias	13/10/2011	17/10/2011	17	Fulano 8
20		RNF8 - Constru	4 dias	13/10/2011	18/10/2011	17	Fulano 4
21		RF227 - Manter	3 dias	18/10/2011	20/10/2011	19	Fulano 3
22		RF299 - Cadast	2 dias	19/10/2011	20/10/2011	20	Fulano 8
23		RF300 - Pesquis	2 dias	19/10/2011	20/10/2011	20	Fulano 4
24		RF312 - Impleme	3 dias	13/10/2011	17/10/2011	17	Fulano 9
25		RF317 - Integra	2 dias	18/10/2011	19/10/2011	24	Fulano 9
26		RF327 - Finaliza	1 dia	21/10/2011	21/10/2011	21;25	Fulano 3
27		**Especificar Teste**	**2 dias**	**11/10/2011**	**12/10/2011**		
28		Elaborar Especi	2 dias	11/10/2011	12/10/2011	16	Fulano 6
29		**Executar Testes**	**2 dias**	**13/10/2011**	**14/10/2011**		
30		Executar Plano	2 dias	13/10/2011	14/10/2011	28	Fulano 7
31		Finalizar Fase de Cons	1 dia?	24/10/2011	24/10/2011	26;30	
32		**Transição**	**2 dias**	**25/10/2011**	**26/10/2011**		
33		**Integrar Sistema**	**2 dias**	**25/10/2011**	**26/10/2011**		
34		Preparação e V	1 dia	25/10/2011	25/10/2011	31	Fulano 2
35		Revisar Projeto	1 dia	26/10/2011	26/10/2011	34	Fulano 2
36		**Disponibilização da**	**2 dias**	**26/10/2011**	**27/10/2011**		
37		Acompanhamento	2 dias	26/10/2011	27/10/2011	34	Fulano 1
38		Teste de Aceitação	2 dias	26/10/2011	27/10/2011	34	Fulano 6
39		**Fechar Projeto**	**1 dia**	**28/10/2011**	**28/10/2011**		
40		Reunião para Encer	1 dia	28/10/2011	28/10/2011	37;38	Fulano 1
41		Finalizar Projeto	1 dia?	31/10/2011	31/10/2011	40	

Figura 98. Projeto para o cenário 4

E.5 CENÁRIO 5

Veja abaixo a descrição das atividades propostas para o cenário 5.

Cenário 5
Participante: Teste Usuário
Download: Clique aqui para fazer download do cenário 5
Fluxos Organizacionais:
- RH_Contratar: Fluxo Organizacional responsável pela contratação de pessoas.
- RH_Treinamento: Fluxo Organizacional responsável pelas atividades de treinamento da equipe.
- Financeiro_Comprar: Fluxo Organizacional responsável pela aquisição de equipamentos.
- Negocios_Parceria: Fluxo Organizacional responsável pela realização de parcerias e contratação de empresas terceirizadas.

Descrição do Cenário 5:
O pessoal mais qualificado está doente e não disponível nos momentos críticos.
Orientações:
1. Abra o arquivo Cenário5;
2. Durante a execução da atividade "Teste da Arquitetura", no dia 05/10/2011, você percebe que os componentes de software adquiridos de uma empresa terceirizada e que seriam reutilizados neste projeto contêm defeitos que limitam sua funcionalidade. Entretanto, esta atividade é necessária para as atividades abaixo de "Especificação Requisitos - Release 1.0". Desta forma, você deve entrar em contato com o setor administrativo informando a necessidade de entrar em contato com a empresa fornecedora do produto. O contato com as empresas fornecedoras é uma atividade pertencente ao fluxo organizacional do setor administrativo, cujo tempo de duração é de 10 (dez) dias.
3. Com base nesta situação, sem alterar a ordem das atividades no projeto, defina qual é o prazo estimado para que a atividade "Especificação Requisitos - Release 1.0" seja iniciada **(Digite sua resposta)**.
4. Caso você esteja usando o modelo SPIM:
   - Atribua uma relação de dependência entre as atividades do projeto de software e as atividades pertencentes ao fluxo organizacional: Clique na atividade que depende do resultado da realização do fluxo organizacional para que seja iniciada. Após, selecione a aba "Campos Personalizados". No campo personalizado chamado "Fluxos Organizacionais", informe o nome do fluxo organizacional (tenha atenção para que tenha exatamente o mesmo nome definido na tabela acima).
   - Ative o filtro listado abaixo e faça a validação do projeto: Associação com os Fluxos Organizacionais.

Salve o arquivo com o nome: Cenário5_SeuNome

Respostas

Resposta: ▪ Arquivo: ▪ Salvar

Veja abaixo uma parte do projeto a ser avaliado no cenário 5.

		Nome da tarefa	Duração	Início	Término	Predecessoras	Nomes dos recursos
15		**Arquitetura**	**3 dias**	**10/10/2011**	**12/10/2011**		
16		Definição do processo de implementação	1 dia	10/10/2011	10/10/2011	13	Fulano 2
17		Modelar Banco de Dados	2 dias	11/10/2011	12/10/2011	16	Fulano 5
18		**Especificação Requisitos - Release 1.0**	**7 dias**	**13/10/2011**	**21/10/2011**		
19		RNF6 - Front End em Inglês - DSV	3 dias	13/10/2011	17/10/2011	17	Fulano 8
20		RNF8 - Construção do Front End - ETR	4 dias	13/10/2011	18/10/2011	17	Fulano 4
21		RF227 - Manter Perfis - ERR	3 dias	18/10/2011	20/10/2011	19	Fulano 3
22		RF299 - Cadastro de Pessoas	2 dias	19/10/2011	20/10/2011	20	Fulano 8
23		RF300 - Pesquisar Produtos	2 dias	19/10/2011	20/10/2011	20	Fulano 4
24		RF312 - Implementar Carrinho de Compras	3 dias	13/10/2011	17/10/2011	17	Fulano 9
25		RF317 - Integrar com sistema de cartão de	2 dias	18/10/2011	19/10/2011	24	Fulano 9
26		RF327 - Finalizar compra	1 dia	21/10/2011	21/10/2011	21;25	Fulano 3
27		**Especificar Teste**	**2 dias**	**11/10/2011**	**12/10/2011**		
28		Elaborar Especificação de Plano de Teste (	2 dias	11/10/2011	12/10/2011	16	Fulano 6
29		**Executar Testes**	**2 dias**	**13/10/2011**	**14/10/2011**		
30		Executar Plano de Teste	2 dias	13/10/2011	14/10/2011	28	Fulano 7
31		Finalizar Fase de Construção	1 dia?	24/10/2011	24/10/2011	26;30	
32		**Transição**	**2 dias**	**25/10/2011**	**26/10/2011**		
33		**Integrar Sistema**	**2 dias**	**25/10/2011**	**26/10/2011**		
34		Preparação e Validação do Ambiente	1 dia	25/10/2011	25/10/2011	31	Fulano 2
35		Revisar Projeto e Aprovações	1 dia	26/10/2011	26/10/2011	34	Fulano 2
36		**Disponibilização da release 1.0 - PRD**	**2 dias**	**26/10/2011**	**27/10/2011**		
37		Acompanhamento do projeto	2 dias	26/10/2011	27/10/2011	34	Fulano 1
38		Teste de Aceitação	2 dias	26/10/2011	27/10/2011	34	Fulano 6
39		**Fechar Projeto**	**1 dia**	**28/10/2011**	**28/10/2011**		
40		Reunião para Encerrar Projeto	1 dia	28/10/2011	28/10/2011	37;38	Fulano 1
41		Finalizar Projeto	1 dia?	31/10/2011	31/10/2011	40	

Figura 99. Projeto para o cenário 5

E.6 QUESTIONÁRIO DE AVALIAÇÃO DO MODELO SPIM

Veja abaixo o questionário de avaliação do modelo SPIM.

Avaliação do modelo SPIM

Resumo sobre os conceitos do modelo integrado:

Ao realizar planejamento de um projeto, o gerente de projetos pode necessitar interagir com outros departamentos da organização a fim de obter informações relevantes sobre as atividades gerenciais de apoio necessárias para a realização das atividades produtivas do projeto. Por exemplo, durante o planejamento de atividades de projeto de software, o gerente de projetos informa ao setor de recursos humanos sobre a necessidade de contratação de um administrador de banco de dados. Neste caso, constata-se a existência de uma relação de dependência entre as atividades do projeto de software (tais como, a modelagem do banco de dados) com as atividades pertencentes ao fluxo de trabalho do setor de recursos humanos referentes à contratação do profissional requerido para executar a atividade de produção.

Assim, percebe-se que o fluxo de atividades de um projeto de software pode interagir com os demais fluxos de atividades da organização (fluxos organizacionais).

Avaliação do SPIM:

1. Com base nas informações vistas nesta palestra, marque abaixo como você percebe os benefícios em fazer o planejamento integrado de atividades gerenciais e produtivas para projetos de desenvolvimento de software de acordo com a classificação apresentada abaixo:

1 – Nenhum   2 – Pouco   3 – Médio   4 – Alto   5 – Muito alto

Restrições Verificadas pelo Modelo Integrado SPIM	Resposta				
	1	2	3	4	5
1.1) Redução do tempo no processo de elaboração do planejamento do projeto;					
1.2) Identificação das dependências entre as atividades gerenciais de apoio e de produção;					
1.3) Identificação e mensuração dos custos indiretos do projeto advindos das atividades gerenciais de apoio;					
1.4) Capacidade de ter acesso a informações dos fluxos organizacionais (pertencentes aos outros departamentos da organização);					
1.5) Capacidade de minimizar distorções no planejamento de projetos (tais como, o aumento dos custos e atrasos nos prazos do projeto) pela desconsideração de que as atividades gerenciais de apoio utilizam recursos não alocados diretamente ao projeto de software;					
1.6) Permite antecipar as necessidades advindas das áreas de apoio da organização durante o planejamento do projeto;					
1.7) Distinção explícita entre as atividades produtivas e gerenciais de um projeto de software com as atividades gerenciais de apoio dos demais departamentos da organização;					

2. Você percebeu outros benefícios observados no modelo integrado SPIM durante a sua utilização neste projeto?

3. Você percebeu algum aspecto que não favoreça a gestão de projetos usando o modelo integrado SPIM durante a sua utilização neste projeto?